BigFoot: Big Data Analytics of Digital Footprints

Project name	BigFoot
Project ID	FP7-ICT-ICT-2011.1.2 Call 8 Project No. 317858
Working Package Number	WP5
Deliverable Number	D.5.1
Document title	An Overview of Datacenter Virtualization Technologies
Document version	1.0
Author	TUB, EUR
Date	5-April-2013
Status	*Public*

Revision History

Date	Version	Description	Author
06/02/'13	0.1	Initial Deliverable Setup	TUB
01/03/'13	0.1	Outlined defined	TUB
15/03/'13	0.2	Work on general background	TUB
27/03/'13	0.5	Work on BigFoot research lines	EUR
02/04/'13	0.5	Document Review	EUR
02/04/'13	0.6	Work on Virtualization and Cloud Computing	TUB
03/04/'13	0.7	Work on Introduction and Conclusion	TUB
04/04/'13	1.0	Document Review	EUR

Executive summary

This deliverable is released after the first six months of the EU project BigFoot, and is related to the first phase "Design and Specification" of the work package WP5.

In this deliverable we present a survey on current practices and techniques that have been proposed to virtualize computation and networking in a datacenter environment. We start with a categorization of datacenters today. We then show how virtualization has become a key enabler of multitenancy and cloud computing in datacenters. We also highlight the many options that are now offered to virtualize networks and servers in the wide-network and distributed datacenter environment. We conclude by sketching promising research directions and elaborating on how the BigFoot approach can address a number of them.

Contents

1 Introduction

Enabled by the successful application of the virtualization design principle, Cloud computing revolutionized and continues to transform industry and our society:

- Computation (and software) has nowadays become a commodity;
- Internet services can be deployed cheaply without large hardware investments, or costs for human operation
- Cloud computing also solves the problem of efficiently provisioning for a service with uncertain popularity, by making it possible to adapt the deployed application capacity on-demand

An extensive overview of the benefits outlined above can be found in [26, 56, 47, 94, 93].

The advent of arbitrarily scalable and elastic resources at low cost is attractive for various services, ranging from time-critical big data analytics to new proximity-aware and latency-critical applications. Such services can be deployed on *public* or *private clouds* both spanning one or more datacenters. Public cloud are based on the pay-as-you-go principle and mainly consists in utility computing; private clouds, instead, live within the premises of and authoritative domain of a single organization.

The purpose of this document is to provide an overview of the literature and state-of-the-art solutions in the area of large-scale distributed computing in virtualized contexts such as the private and public clouds. A main emphasis will be on the existing hardware architectures as well as the ability to scale resources elastically.

The remainder of this document is organized as follows. Section 2 overviews the different types of datacenters that exist today. Section 3 is dedicated to a general overview on how datacenter resources can be shared among multiple tenants. Section 4 focuses on the role of virtualization as a key enabler for cloud computing.

A particular emphasis is also put on the networking part, as it constitutes a major subject of investigation in the BigFoot project. Section 5 is based on the general literature surveyed before and discusses additional recent works that are closely related to the vision we have in BigFoot. In particular, we discuss a number of relevant open issues that we will study during the course of the project. We here focus on an experimental and measurement-based approach on prototype private clouds we develop in BigFoot.

2 Datacenter Types

Depending on business needs, there is a long spectrum of datacenter types. In this section, we present a basic classification of datacenters with respect to requirements such as availability, operation and deployment models and approach to multi-tenancy.

2.1 Availability Requirements

Datacenters may have a wide range of availability requirements. Availability of a service is driven by the criticality of the services that are being provided through it. The Uptime Institute [18] defines multiple tiers for datacenter site availability on the dimensions of: downtime per-year, redundancy in the distribution path for power and cooling, as well as susceptibility to disruptions from planned and unplanned activities. We now describe the tiers.

Tier I - Basic Site Infrastructure: the site has non-redundant power and cooling components, as well as a single, non-redundant distribution path. The site is susceptible to planned and unplanned disruptions through operational errors by administrators, or outages of the capacity components or distribution path. The site has to offer 99.671% availability (downtime of 28.8 hours per year).

Tier II - Redundant Site Infrastructure Capacity Components: the site has redundant capacity components (power and cooling), but only a single distribution path. This implies that bringing down distribution path components for maintenance would cause a disruption of the services hosted on the datacenters servers. The site has to offer 99.741% availability (downtime of 22.0 hours per year).

Tier III - Concurrently Maintainable Site Infrastructure: the site has redundant capacity components. There are multiple distribution paths: one active and one in standby. The improved redundancy allows conducting planned maintenance without disrupting services. However, unplanned events will still cause disruption. The site has to offer 99.982% availability (downtime of 1.6 hours per year).

Tier IV - Fault Tolerant Site Infrastructure: the site has redundant capacity components and multiple, simultaneously active distribution paths. This increases fault tolerance of the site in light of component failures, distribution path failures, or planned maintenance. The site has to offer 99.995% availability (downtime of 0.4 hours per year).

2.2 Operation and Deployment Models

Depending on the business use-case, there are multiple operation and deployment models for datacenters today:

2.2.1 Private Datacenters

Private datacenters are purpose-built datacenters for the specific needs of a corporation. The infrastructure is thus operated and used by the same organization, within the same authoritative domain: nevertheless, multiple tenants (e.g., different teams, departments or projects) may be granted cloud resources.

These datacenters often serve to accommodate a mix of development and production deployments of storage and compute platforms. Much of the existing literature has focused on optimizing different layers, in particular datacenter networking: [24, 50, 51, 61, 25, 78, 77, 33, 62, 34, 49, 104].

The increasing popularity of data-intensive open-source applications (e.g. Hadoop [6]) to perform data analytics, is driving the deployment of private datacenters. Prime operators of such datacenters are financial companies, large corporations, web service providers, just to name a few. The exact number of such datacenters is unknown, but likely to be significant. Furthermore, a large number of scientific computational clusters and datacenters hosted in universities and research centers also contributes to the known[1] infrastructure.

2.2.2 Public Datacenters

Datacenters are also deployed to satisfy local as well as global markets. A rough estimation of the currently operated small to medium size datacenters accounts more than 2280 datacenters in around 84 countries [7] and most of them offer servers, storage and network resources for lease. Many of them offer dedicated or virtualized servers, or both. Typical services that are offered are Web hosting and storage. For example, Softlayer has more than 191.5K servers in 7 datacenters [99], OVH has more than 120K servers and more than 3.5K cloud servers [82], 1&1 has more than 70K servers in 5 datacenters [20], Leaseweb has more than 50K servers in 6 datacenters [69] and Rackspace has more than 190K customers in 8 datacenters [89]. By far the most successful story in leasing virtualized resources using the public

[1]In the sense that details on the architecture are publicly available

cloud is Amazon Web Services (AWS) [95]. Amazon maintains 7 large datacenters and caches in at least 25 locations around the world for its CDN solutions, CloudFront. In 2009 AWS operated only 40K servers and by the end of 2012 the number of servers is estimated to exceed 400K [76].

2.2.3 Distributed Datacenters

Despite the economies of scale that a centralized datacenter offers, there is an increasing trend to build datacenters and interconnect them. Apart from the above-mentioned distributed datacenters, Content Distribution Networks (CDNs) are massively distributed infrastructures [70] continuously deployed to cope with volatile and increasing demand for content, which needs to be replicated and made accessible from different locations in the Internet [58, 101, 23]. Some CDNs place their servers deep inside the network whereas others that provide good connectivity to eyeball networks use also Internet Exchange Points (IXPs) [22]. Akamai operates more than 120,000 servers in more than 2,000 locations across nearly 1,150 ISP networks [81, 3]. Akamai utilizes a number of techniques to improve content delivery and application acceleration [70]. These techniques include: assembly of pages on the fly and at the edge, prefetching, compression and delta encoding, and monitoring of the state of the network and path diversity, to name a few. Google is reported to operate tens of datacenters and front-end server clusters worldwide [64, 100, 10]. Microsoft has deployed its CDN infrastructure in 24 locations around the world [29]. Limelight operates thousands of servers in more than 22 delivery centers and connects directly to 600 networks worldwide [79]. Distributed datacenters are also used for delivering bulk data over then Internet [68].

2.3 Tenancy in terms of Infrastructure

The cost of deploying, maintaining, managing and replacing infrastructure is high [47]. To that end, different management solutions have been applied to better allocate infrastructure resources, from fully dedicated to shared.

2.3.1 Co-location

Co-location is a term coined to describe the hosting of infrastructure off-site: it emerged from the need to install and operate hardware, including routers and frontend servers, in multiple and diverse geographical and network-wise locations. Large companies have evolved, such as Equinix [8] that is located in 95 points and 15 countries around the globe. IXPs [22] also facilitate co-location centers — today, there are more than 320 IXPs [9] deployed in major cities. Co-location centers typically offer the choice of leasing hardware, placing hardware, as well as leasing slices of shared resources.

2.3.2 Dedicated Infrastructure

For many applications such as video streaming the full utilization of the computational, storage and network resources are needed. Today, a large fraction of the Internet traffic is due to video [59, 66, 46, 43]. To cope with this demand, large online Video providers such as YouTube and Netflix have developed their own datacenters or rely on large CDNs [21]. Moreover, such large content providers deploy their own servers inside consumer ISPs to reduce inter-domain traffic and to improve end-user performance. Google has launched Google Global Cache [11, 28], partnering with ISPs to optimize network costs associated with Google traffic, especially video traffic from YouTube. Netflix, being currently responsible for more than 30% of the traffic in North America [59], has introduced Open Connect CDN [13] with similar goals. Both Google Global Cache and Netflix Open Connect appliances are located inside ISPs or, alternatively, in interconnection and peering points.

2.3.3 Shared Infrastructure

Very recently, a number of large network operators published a white paper to describe their view on network function virtualization [19]. Network function virtualization constitutes a paradigm shift of deploying service-specific appliances inside the network to transform it into an open and generic-usage appliance; such services include message routers, carrier-grade NATs (Network Address Translation), broadband remote access server (BRAS), deep-packet inspection systems, firewalls, load balancers, to name a few examples.

Such generic appliances are physically built out of standard high volume servers, storage and Ethernet switches. Independent software vendors can install their applications in these generic appliances, located deep inside the network or in datacenters. This setting leads to a win-win situation to different parties that are involved in the deployment and operation of server infrastructure [83, 84, 85]. For example, Akamai recently announced the formation of content delivery strategic alliances with major ISPs, namely AT&T [1], Orange [2] and Swisscom [16]. Such alliances allow, for example, both CDNs and ISPs operators to **jointly** deploy server infrastructures within the network to reduce cost, improve end-user experience, improve traffic engineering. Such deployment can use bare metal solutions or take advantage of virtualization.

The management of computation, storage and network components, in addition to the task of orchestrating the deployment of virtual machines (VMs) on top of physical servers is today possible through "cloud-management systems". The default interface that is currently advocated by many projects is Openstack [14], which is supported by a number of vendors and software houses. OpenStack is is also widely used in virtualized datacenters. In addition, there are solutions tailored to generic appliance provision and management such as SCC [74].

Theoretical results have also shown that sharing of computation and network resources can be done in an online fashion in wide area networks [113] as well as in datacenters [63]. Last but not least, recent studies have also shown that outsourcing service functionality can significantly reduce the operational cost of enterprises without significantly sacrificing performance [96].

2.4 Tenancy in terms of Application Mix

Benson et al. [32] present a detailed study of network traffic characteristics of universities, enterprises and cloud datacenters. It highlights important differences between utilization of these datacenters with regards to the kind of applications being used, traffic patterns and link-utilization at different layers of the topology. Two of the datacenters that were surveyed were **single-purpose** cloud datacenters whereas the others were **multi-purpose** enterprise and campus datacenters. We now describe characteristics observed by the authors that are relevant for this survey.

Non-uniformity of application placement: Different datacenters have very different mixes of applications running in them, with placement of applications across different physical servers being non-uniform. This leads to non-uniform traffic combinations being observed across different switches in the datacenter. This heterogeneity would be inherently more prevalent in virtualized and hosted environments due to a larger number of hosted applications.

Multiple inter-dependent components: In all datacenters that were surveyed, there was either a mix of inter-dependent applications or components within a single application that was distributed. For instance, in the surveyed university and enterprise datacenters, traffic to authentication services from web-portals were observed. The university datacenters also had a heavy traffic share attributed to the distributed file systems in use. In three of the cloud datacenters that were surveyed, the applications being run were typically composed of multiple applications with intricate dependencies, deployed across the datacenter. An example of such an interaction is that of a social networking web-site's frontend requiring access to authentication services for verifying users and different sets of data-stores for aggregating different kinds of data. Two of the cloud datacenters were dedicated to running MapReduce jobs.

Network communication patterns: Two of the cloud datacenters that were surveyed were used primarily for running MapReduce jobs, while the other three hosted a mix of customer-facing applications and the backends to support them. In all these cases, the authors observed a high degree of intra-rack traffic with at least 75% of traffic confined to the rack in which it was generated. This is attributed to co-location of inter-dependent applications. However, in the case of the private and educational datacenters that were surveyed, at least 50% of the traffic generated at the servers was observed to leave the rack. A possible reason for this is poor co-location of dependent applications. The communication patterns in the cloud datacenters also displayed time-of-day and week-of-day traffic patterns.

Flow level patterns: The authors observe that for the private and educational datacenters that were surveyed: 1) the number of active flows per-switch in any given second is at most 10,000; 2) new flows may arrive within 10μs of each other; 3) most flows were small in size (less than 10KB), of which, a significant fraction are only a few hundreds of milliseconds long; 4) traffic leaving the edge switches are bursty in nature with ON/OFF intervals following a heavy tailed distribution; 5) The predominant application in some datacenters drove the aggregate sending pattern at the edge switch. The authors also point out that it is difficult, in the general case, to understand the characteristics of packet transmission in the datacenter network only based on the dominant applications.

2.4.1 Single-purpose

Single-purpose datacenters are those that are built with a single application in mind. This is particularly prevalent today in super-computing and data-analytics clusters, where virtualization is not desirable due to performance overheads. There is a rich literature [24, 50, 51, 61, 25, 78, 77, 33, 62, 34, 49, 104], that studies the architecture and operation of such single-purpose datacenters, e.g., the kind operated by Microsoft, Google and Facebook.

2.4.2 Multi-purpose

With regards to application mix, multi-purpose datacenters exhibit greater degree of heterogeneity unlike the single-purpose case. Infrastructure is often virtualized, with multiple tenants deploying applications atop the same infrastructure. Examples of such settings include Amazon's AWS cloud offerings and traditional hosting centers [95]. Large telecommunication providers, such as AT&T and Deutsche Telekom operate large datacenters and offer hosting solutions in their corporate customers. Recently such infrastructures have been virtualized upon strategic alliances with virtualization leaders such as VMWare [17]. Unfortunately, most of the published work in the literature focuses on single purpose datacenters. Thus, the operation and characteristics of multi-purpose datacenters is largely unknown.

3 Virtualization as an Enabler of Multi-tenancy

Virtualization is arguably the main innovation motor for the future Internet. Today's datacenters are already highly virtualized and we expect that the virtualization trend will soon spill over to Internet as well. A particularly interesting paradigm in this context is *network virtualization*: it envisions an Internet where arbitrarily specified *virtual networks (VNet)* can be requested on demand and at short notice. Multiple VNets can share a given physical infrastructure but provide the illusion of a dedicated network; thus, resource efficiency can be improved. Moreover, VNets can be *tailored* to the specific application and implement their own layer 3+ protocol stack.

At the heart of virtualization lie the ideas of (1) making more efficient use of a given infrastructure by sharing it among multiple tenants while providing isolation; (2) decoupling resources and services from the constraints of the underlying physical infrastructure; and (3) abstracting heterogeneous hardware and provide a unified and simplified view of the resource network.

3.1 Compute Virtualization

Compute virtualization has revamped the server business over the last years. Nowadays, an almost unbounded amount of computing power can be dynamically leased in different datacenters. Hardly any cloud provider today offers physical machines anymore: rather, the supplied machines are virtual (virtual machines VM). This trend heralded the new paradigm of *elastic computing*: virtual machines can be allocated and deallocated on demand and at short notice. The resulting flexibilities and the dynamic resource scaling are one of the main advantages why such virtualized compute architectures are considered in the context of big data analytics.

3.1.1 Server Virtualization

In server virtualization, multiple operating systems are run atop a virtual machine monitor[2] (VMM), enabling better utilization of physical resources available in the data-center. This architecture lends itself to the following three flavors:

Native VM system: The VMM operates in a privilege mode higher than the guest VMs, which is typically the highest privilege level defined by the system architecture. The guest OS' privilege level is emulated by

[2]Virtual machine monitors are often referred to as *hypervisors*.

the VMM. Since the VMM runs directly atop hardware, this mode of virtualization offers higher performance. Xen [87] is an example of such a virtualization technology.

User-mode hosted VM: In this mode, the VM system is installed on the host platform atop an already existing OS. The VMM makes use of the host OS' features to perform resource management of the VMs. There are no special drivers required for such a system, but this leads to a performance penalty. Virtualbox [102] is an example of such a technology.

Dual-mode hosted VM: The VMM operates partly in privileged mode and partly in non-privileged mode, leading to benefits offered by both the above methods of system virtualization. This normally requires kernel extensions and specific device drivers for the host OS. E.g., QEMU [88] with KVM [65].

While these methods discuss the position of the VMM with respect to the hardware and privilege levels, we now discuss virtualization techniques from the guest VM's perspective. In **full-system virtualization**, guests VMs run unmodified atop the host operating system. Virtualbox is an example of such a technology. In **Para-virtualization**, the guest VMs are "virtualization-aware". For instance, with Xen, some privileged instructions executed by the guests are replaced with "hypercalls", with multiple hypercalls being batched for efficiency. The advantage here is not only performance gains, but also reduced complexity at the hypervisor layer. For instance, the hypervisor does not need to perform complex code detection/discovery, code patching nor shadow table maintenance.

3.1.2 Migration

The decoupling of compute services from the underlying infrastructure, as it is introduced by virtualization, also enables the flexible migration: of virtual machines or entire virtual networks (VNets). There are two main reasons why migration support may be relevant in the context of big data analytics: (1) the dynamic resource scaling may require re-configurations, e.g., as more resources are needed or because resources are freed up: it can make sense to collocate frequently communicating VMs dynamically; (2) the properties of big data as well as the resulting communication patterns (e.g., the shuffle phase in a MapReduce job[3]) may not be known in advance and only crystallize out during the computation. In this sense, computing VMs or traffic flows may be migrated to the location where they are most useful (e.g., to collocate with the data or avoid highly-loaded paths).

[3]See deliverable D.2.1 for details on MapReduce.

But migration can also be used in the wide-area context to improve latency and other QoS parameters. Concretely, imagine a scenario where an ISP uses network virtualization technology to offer an SAP or a *game server* service to mobile, thin-clients. If the service request patterns change over time, e.g., due to commuting users, due to timezone effects, or due to unexpected events in sports or politics, rendering it worthwhile to migrate the service to different locations. For instance, it can make sense to transfer a service from China to Europe at night, to improve the access to the service both in terms of latency as well as cost (e.g., due to roaming) [53].

While migrating services may improve access latencies, migration comes at a cost. For example, the bulk-data transfer imposes load on the network and may cause a service disruption. One of the main parameters influencing this cost is the available bandwidth along the migration path in the substrate network [38]. If virtual networks (*VNets*) are provisioned across administrative domains belonging to multiple infrastructure providers, (inter provider) migration can also entail certain transit (or *roaming*) costs.

To the best of our knowledge, [35] and [27] (for online server migration), as well as [30, 44] (for online virtual network embeddings) are the only works studying network virtualization problems from an online algorithm perspective. The formal competitive migration problem is related to several classic optimization problems such as facility location, k-server problems, or online page migration. All these problems are a special case of the general *Metrical Task System* (e.g., [36, 37]) for which there is, e.g., an asymptotically optimal deterministic $\Theta(n)$-competitive algorithm, where n is the state (or "configuration") space; or a randomized $O(\log^2 n \cdot \log\log n)$-competitive algorithm given that the state space fulfills the triangle inequality. The randomized algorithm proceeds via a (well separated) tree (or *ultrametric*) approximation for the general metric space (in a preprocessing step) and subsequently solves the problem on this distorted space.

The authors in [67] developed a scalable and fully distributed approach to migrate servers based on the source of demand and cost of locating a server in different locations in the Internet. The proposed distributed approach achieves performance under various synthetic and real Internet topologies and workloads that are comparable to that of optimal, centralized approaches requiring full topology and demand information.

3.1.3 I/O Virtualization

There is a number of reasons why decoupling of a VM's logical input/output (I/O) from the physical implementation can be beneficial [103]. First, it offers the ability to the administrator of a system to multiplex many VMs on the same hardware. This allows public clouds such as Amazon AWS [95] to install a number of VMs in a single physical server. Second, it allows the live migration of running VMs. Third, it enhances security, e.g., by running an encryption function over the I/O to and from a disk to implement disk encryption. A recent study [108] of different virtualization solutions, namely, Linux, VServer and Xen, shows that under intense I/O load, significant packet delay is added under virtualization. Thus, despite the fact that virtualization may increase hardware efficiency, it can have significant effect on the network performance of an application. In [106], the authors also showed that the network performance can be reduced even under low I/O load in shared environments.

3.2 Network Virtualization

There has been a significant interest in virtual and cloud networks over the last years. The reader is referred to the recent surveys [40] and [52].

The network virtualization paradigm takes the virtualization trend one natural step further: rather than only virtualizing the nodes, the links between the VMs are also virtualized and provide resource guarantees. This is necessary especially for time-critical big data analytics which need to meet deadlines, and where link resources constitute the bottleneck. Essentially, there is not point of having virtualized resources if they cannot be accessed in a timely manner!

The paradigm revolves around the abstraction of a virtual network (VNet): a graph where vertices represent cloud resources and links are the connections (e.g., VLANs, MPLS, OpenFlow, ...). The main challenge of embedding a VNet to compute big data is algorithmic: The survey by Belbekkouche [31] provides a nice overview of allocation and embedding algorithms. Most of the algorithms proposed in the literature today are heuristics and come without any formal performance guarantees. For example, Fan and Ammar [45] study dynamic re-configurable topologies to accommodate communication requirements that vary over time, Zhu and Ammar [113] consider virtual network assignment problems with and without reconfiguration but only for bandwidth constraints, Ricci et al. [90] pursue a simulated annealing approach and Lu and Turner [73] seek to find the best topology in a family of backbone-star topologies. Many approaches in the literature fail to exploit the flexibility to embed virtual nodes and links simultaneously and solve the two mappings sequentially (e.g., [72]), which entails a loss of efficiency [41]. To deal with the computational hardness, Yu et al. [112]

advocate rethinking the design of the substrate network to simplify the embedding, e.g., by allowing to split a virtual link over multiple paths and perform periodic path migrations. The focus of the work by Butt et al. [39] is on re-optimization mechanisms that ameliorate the performance of the previous virtual network embedding algorithms in terms of acceptance ratio and load balancing; their algorithm is able to prioritize resources and is evaluated by simulations. Virtual network embeddings have also been studied for cross-provider settings [110] and from a distributed computing perspective [57].

3.2.1 Software-defined Networking

Network switches and routers are architecturally composed of two components: a *data plane* and a *control plane*. The data plane handles forwarding of packets that arrive at the device and is thus also known as the *forwarding plane*. The control plane handles the logic needed in order to correctly set up a forwarding plane (that is, correctly set up a set of forwarding rules) at the switch or router.

Traditional switches and routers have proprietary firmware and control logic implementations. This not only inhibits vendor interoperability, but also hampers flexibility. Even though these switches can be configured or managed through Simple Network Management Protocol (SNMP) or a command line interface (CLI), there are no means of introducing a new control plane function or protocol into the switch. This makes experimenting with new networking protocols cumbersome.

The Software Defined Networking (SDN) approach aims to alleviate these issues by decoupling the data plane from the control plane. The OpenFlow [75] protocol is largely considered an enabler of SDN; it dictates a protocol for a remote controller to manipulate the forwarding tables of a switch. In an OpenFlow based architecture, a logically centralized network controller speaks OpenFlow to a network of switches. Third-party software can then orchestrate a network using interfaces exposed by the OpenFlow controller.

SDN research has had significant focus on data-centers. For instance, Hedera [25] is an adaptive flow scheduling system for data center networks. The premise of Hedera is that existing IP multipathing techniques used in data centers usually rely on per-flow static hashing, which can lead to under-utilization of some network paths over time due to hash collisions. The systems detects large flows at the edge switches of a data center and uses placement algorithms to find good paths for the flows in the network.

In [107], the authors implement server load balancing using OpenFlow switches. The authors observe that the number of flow entries that can be saved on an OpenFlow switch is much less than the number of unique flows that a switch might need to handle in data center workloads. Thus, micro flow management using per-flow rules is not practical for distributing flows between different servers using a switch. The authors thus take advantage of OpenFlows wildcard based rules capability and propose algorithms to compute concise wildcard rules that achieve a specic distribution of trac.

DevoFlow [42] highlights the bottleneck at the OpenFlow switch within the context of high-performance networks (like data centers). This occurs because in OpenFlow based networks, if a packet is received for which there is no matching forwarding rule, the packet is forwarded to the controller. Assuming the control logic works only reactively, the controller will end up handling the first packet of every flow before inserting a rule. For a switch to send a packet to the controller, it has to first move the packet from hardware to its management CPU. This can be a bottleneck within high performance networks, where there are several hundred thousand flows being generated per-second. The authors propose to tackle the problem by addressing short-lived (mice) and long-lived (elephant) flows separately. Switches only inform the controller about those flows which have lasted beyond a threshold.

3.2.2 Virtual Networks

Virtual networks (VNets) may be implemented in different ways. While the node virtualization part can be realized using VMWare technology, the links can be realized with (e.g., SDN, VLANs). There also already exist unified frameworks, e.g., the OpenStack framework.

A BigFoot partner, TUB, has implemented a prototype to realize virtual networks connecting cloud resources, henceforth called *CloudNets* (short for *cloud networks*).[4]. It combines node and link virtualization and offers Quality-of-Service (QoS) guarantees, both on the nodes and the links. Basically, a CloudNet describes a virtual network topology where the virtual nodes represent cloud resources (e.g., storage or computation) which are connected by virtual links.

While CloudNet concepts are already emerging in the context of data centers (e.g., [109]), we expect that in the future, Internet Service Providers (ISP) will also offer flexibly specifiable and on-demand virtual networks, connecting (heterogeneous) cloud resources with connectivity guarantees.

[4] `http://www.net.t-labs.tu-berlin.de/~stefan/virtu.shtml`

Such an *elastic* wide-area network (WAN) connectivity is attractive in many settings. For example, for inter-site data transfers or state synchronization of a distributed application (such as online gaming). Other use cases are the *spill-over* (or *out-sourcing*) to the public cloud in times of resource shortage in the private data center, or the distribution of content to CDN caches.

Note that a CloudNet may not even specify the locations of its constituting resources; hence, the mapping of the CloudNet can be *subject to optimization*. In fact, the resources of the CloudNets can be *migrated* over time. For example, latency-critical CloudNets (e.g., realizing a game, an SAP or a social networking service) can be dynamically migrated closer to the users, while delay-tolerant CloudNets (e.g., for large-scale computations or bulk data storage) are run on the remaining servers. Moreover, resources allocated to a CloudNet can be scaled up or down depending on the demand at the different sites. Finally, the decoupling of the CloudNet from the underlying physical infrastructure can also improve reliability, as networks can seamlessly switch to alternative cloud and link resources after a failure, or for maintenance purposes (see e.g., the Amazon outage in April 2011[5]).

Our CloudNet architecture encompasses the Physical Infrastructure Provider (PIP) role and the Virtual Network Provider (VNP) role. The two roles communicate requirements and allocations via clear negotiation interfaces and using a generic resource description language. Since interfaces between players are generic and since we do not distinguish between physical and virtual resources, a recursive role concept is supported (e.g., a VNP sub-structured into other VNPs).

Moreover, a high generality is achieved by a *plugin architecture* which allows for replacement of underlying technologies and operating systems.

[5]`http://aws.amazon.com/message/65648/`

4 Virtualization as an Enabler of Cloud Computing

Since virtualization enables multi-tenancy, it is at the heart of many cloud computing business models today: such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS).

4.1 Infrastructure as a Service

In the IaaS model, the cloud provider offers customers the abstraction of a full operating system and hardware resources. More often than not, the provided infrastructure is virtualized through a system virtualization technology such as Xen [87] or KVM [65]. Most IaaS vendors also provide services such as image libraries, object/block/file storage services, load balancers, policy based access control and virtual networking services. This saves the customer the overhead of designing and maintaining their own data-centers and hardware deployments.

Examples of IaaS providers are Amazon EC2, Google Compute Engine and Rackspace.

4.2 Platform as a Service

In a PaaS, the provider presents a programming language execution environment and necessary storage layers to the user, while abstracting away physical machines and their actual operating systems completely. The customer does not deal with maintaining any software or hardware that comprise the underlying stack. From the customer's point of view, one of the key advantages of the PaaS model is that the underlying execution environment can automatically scale the application up or down, without the customer having to provision resources explicitly for it.

Example PaaS providers are Heroku and Google App Engine.

4.3 Software as a Service

In this model, the user uses application software that is installed, deployed, and managed remotely by the cloud provider. The user need not be concerned with deploying the application on her own infrastructure, thus saving the user from maintenance and support of the software itself. As with PaaS, the provider automatically scales the application depending on the load without requiring the customer's intervention.

Google Apps and Microsoft Office 365 are examples of software delivered through the SaaS model.

5 Open Issues and BigFoot

While virtualization is a stable and widely used technology in datacenters, there are a number of open issues that we will consider in BigFoot.

5.1 Beyond Best-effort Services

The vision behind the BigFoot project is that of offering elastic deployments of large-scale data management systems for the storage and analysis of massive amounts of digital footprints. This is achieved building on private clouds – that is, cluster of server-grade machines dedicated and internal to an administrative domain, e.g. an individual company – for the provisioning of resources (compute, storage and network) to data-intensive services. Resource elasticity enables flexible application performance — an application can be allocated more or less cloud resources to match its performance needs.

Currently, a handful of public services partially address the problem of offering data storage and analysis as a service, such as Amazon's Elastic MapReduce [5]. However, despite resource elasticity offered by todays cloud providers being a key driver for cloud adoption, currently, cloud providers expose a minimal interface to its users or tenants. For a service to be instantiated, tenants are asked to specify the number and "flavor" of compute instances they require; this selection determines service charges, which are computed on a pay-as-you-go basis.

While simple and elegant, prior works [55, 60] note that there is a disconnect between the resource-centric interface currently exposed by providers and what tenants actually require. Namely, tenants are primarily interested in predictable performance for their applications. In addition, costs are another key metric which tenants wish to optimize — this is a problem that is relevant mainly for public clouds that require tenants to pay for their services.[6] For instance, tenants want to satisfy constraints regarding their application completion time: in the context of BigFoot, this translates in how long a, say, batch-processing data analysis will take to complete. With todays setup, tenants bear the burden of translating their application-level performance goals into the corresponding resource requirements, prior to submitting their request for a service.

[6]While pricing cloud services is an interesting topic per se, in this Section we gloss over the design and implementation of pricing schemes, as BigFoot is conceived as a service running on private clouds, which are internal to organizations and that are not subject to costs for its users.

With the above in mind, both related works and BigFoot argue for refactoring the tenant-provider relationship to improve cloud usability: instead of todays resource-centric interface, providers should offer an application-centric interface whereby tenants only specify high-level objectives for their jobs and applications. Such an application-centric interface requires the cloud provider to map tenant goals into resource requirements. While burdening the provider, an application-centric interface also opens up opportunities for providers to optimize the inner resource allocation mechanisms implemented in their cloud management systems to meet applications demand, while at the same time optimizing metrics that are important from the provider perspective, such as load, energy consumption and others.

In the BigFoot project, we consider cloud management systems in the open-source domain – precisely, we use and work on the OpenStack [14] cloud management system. Although not on OpenStack explicitely, recent literature – which we review next – presents some interesting problems that arise at the cloud management layer. For example, the work in [55, 60] consider the problem of creating a "language" to specify application resources in terms of compute and network bandwidth (at the moment, storage is not considered). Then, the problem consists in enumerating all possible resource configurations that may satisfy application needs and pick the one that satisfies at the same time the provider objectives. This is achieved in the particular setting of MapReduce applications, by proceeding with a preliminary profiling phase of data analysis jobs, to infer its service time, for a given configuration. While this approach is in line with the BigFoot project vision, more work has to be done to factor in the performance bottlenecks inflicted by virtualization techniques when computing application performance for each possible resource allocation. As discussed earlier in this deliverable, virtualization (both at the compute and network level) hinders the task of appropriately measuring performance, which may fluctuate unexpectedly and that in general does not match that of "bare-metal" application deployments. In addition, in BigFoot we consider the additional constraints of defining mechanisms that are compatible with OpenStack and, more importantly, with the Software Defined Networking layer of OpenStack, namely Quantum [15]. The latter requirement calls for more attention than simply following the design guidelines of OpenStack: Quantum allows the creation of purely software networks for each tenant in a shared private cloud and comes with several challenges both at the forwarding plane (how fast packets can move on software networks) and at the routing plane (how packets traverse network to reach their intented destinations).

The problems related to system virtualization outlined above – despite the benefits inherent to system virtualization per se – could be circumvented by defining **specialized** private cloud deployments. Essentially, as discussed in [54], the idea is to isolate application performance through a lightweight alternative to virtualization (e.g. Linux Containers [12]) and allocate bare-metal resources to applications using a fine-grained definition of such resources. For example, the work in [54] illustrates the design of Mesos, an orchestration framework to deploy data-intensive frameworks on top of real-hardware available in a cluster. In practice, Mesos is a resource scheduler that negotiate resources with applications and grant a share of cluster resources to meet application needs. In BigFoot we also consider lightweight alternatives to virtualization (precisely, Linux Containers as they are supported by the OpenStack framework), but we do not require applications to be modified or patched to be operational in the cluster; instead, Mesos requires such modifications which we believe could hinder its adoption. In addition, we note that our measurements on a medium-size cluster operated using OpenStack with Linux Containers has the very same network performance bottlenecks (which are dominant for data-intensive applications) as a deployment with traditional virtualization. A promising research direction is to focus on improving current software switches and their impact on performance. In addition, recent work [86, 98] also consider problems related to resource sharing: when a datacenter network is shared among multiple tenants, great zeal has to be dedicated to a proper and fair allocation of network resources, which are not elastic as compute and storage resources.

Next, we present in more details, two research lines that we will develop in the project.

5.1.1 Towards Application-aware VM Placement

In a typical virtualized environment, VMs are placed in a way to increase the utilization of physical resources such as CPU, storage, or network [113, 63]. Thus, the placement is not fully aware of the application that is executed as well as the parallel execution of other applications utilizing neighboring VMs. In every application it is imperative to efficiently allocate resources in a way that the requirements of the service are respected and the execution of the complicated tasks is guaranteed to the end. To that end, fast algorithms to map the requirements as well as the constraints of applications run over on the virtualized environment. In BigFoot, new embedding of online algorithms will be developed tailored to the operation of data-intensive applications like map-reduce.

5.1.2 Software Switch Performance

A recent work that we deem relevant in the context BigFoot, is a recent effort by L. Rizzo *et. al.* [92]. In their work, as we do in BigFoot, the authors acknowledge that Virtual machines need to communicate and access peripherals, which for systems used as servers mostly means disks and network interfaces: depending on application running in virtual machines (that are part of a cloud deployment), disk and network access may be frequent and heavy to handle; this is exacerbated in the context of data-intensive applications that we target in BigFoot. Managing network access is extremely challenging to deal with even in non-virtualized environments, due to the high data and packet rates involved and the fact that, unlike disks, traffic generation is initiated by external entities on which the receiver has no control. It is then not a surprise that virtual machines may have a tough time in operating network interfaces at wire speed in all possible conditions. Simply put, when virtual machines communicate (even with such traffic being limited to an **individual physical host**), the network layer is problematic.

Current solutions to the problem require hardware assistance to improve performance. Some proposals, for example, rely on multi-queue network cards (hardware) exporting resources to virtual machines through PCI passthrough and/or on external switches to copy data between interfaces. In practice, the network used to communicate among virtual machines residing on the same physical host becomes a physical network. Clearly, this solution is expensive and not necessarily scalable.

However, **software-only** solutions proposed to date tend to have relatively low performance: for example, our preliminary measurements on the BigFoot platform – using current best practices for software defined network components – indicate that inter-VM communication on the same physical host is at best in the same order of magnitude that what can be obtained for the same traffic being sent over a physical network. In this context, the work in [92] suggests a novel approach based on memory mapping, which considerably improves upon current solutions: essentially, they propose a new kernel module which implements a software switch that maintains a shared memory across processes that communicate over the switch. Then, data moves in memory at a fast rate.

In BigFoot, we will first consider the work in [92] and understand to which extent it is possible to modify it to be compatible with a full fledged network virtualization control plane (as a reminder, we use the Quantum module of OpenStack). Our objectives are, using the insights gained by performing a thorough measurement campaign, to understand what are the fundamental bottlenecks that current solution have, to understand whether a software based solution that use memory mapping is amenable to a private cloud deployment and propose alternatives to the current state of the art.

5.2 Performance Evaluation of Cloud Services

In the past years, the research community involved in the Networking domain dedicated a great effort into building measurement infrastructures for establishing the performance of current cloud deployments, that rely on a range of virtualization techniques.

Essentially, there are two lines of research that are related to the research activities of BigFoot:

- *Measurements on public clouds*: in this Section we review only a few, noteworthy, articles that focus on establishing the networking performance of public clouds, such as Amazon Web Services and in particular of the Elastic Compute Cloud [4]. As a general remark, such measurement works aim at finding anomalies, with respect to bare-metal clusters with no virtualization, in basic network performance metrics such as throughput and round-trip-times (RTT). However, such works do not pinpoint at the root causes of such anomalies, nor propose solutions to overcome bottlenecks.

- *Measurements on small scale testbeds*: some works, focus on a small deployment, typically striping off cloud management systems and software defined network components. The goal of such work is to focus on an individual performance metric and study any degradation that occurs and that can be attributed to virtualization

First, we discuss works that measure network performance in public clouds [105, 111]. This line of research consists in setting up a number of virtual machines in a public cloud, such as Amazon Web Services and measure end-to-end performance among source/destination pairs selected according to an heuristic (usually, randomly). Measurements are obtained with legacy UNIX command line tools such as `iperf`, `ping` and so on. It is important to observe that, since such measurement campaings are launched in a public cloud, the measurement probes (and hence the interpretation of the results) are not informed about the underlying cloud utilization. For example, on the same physical host running an instance of a measurement probe, another virtual machine operated and owned by a different tenant could use shared resources and hence have an impact on the measurement. In addition, the configuration details of public clouds are generally not disclosed by companies running such services. As such, it is very hard to guess the root causes behind measurement results.

Instead, works such as [108], consider a small scale setting, which is completely under control and dedicated to a measurement experiment. As observed in [108], several important networking metrics exhibit some anomalies, such as fluctuations, instability, bias, which makes them not suitable for applications running in clouds to operate correctly. As an example, applications requiring a stable and unbiased measure of RTT, may encounter problems when run in a cloud environment.

In BigFoot, as defined in WP2, we set an important goal concerning an appropriate benchmarking (using measurements) of the platform used to run the BigFoot software stack. As such, we also have a fully instrumented setup, which is completely under our control (from hardware to software), that we use to inform our research effort into identifying the potential bottlenecks introduced by virtualization (although we study I/O in general, not only network) and to propose solutions to such problems. However, as opposed to prior work, we consider a full fledged deployment including a cloud management system (and its overhead), a network virtualization framework and software switches. In addition, we perform measurements with and without background traffic generated by applications running concurrently on the BigFoot private cloud. Essentially, our goal is to obtain the best of both worlds: the scale and issues that can be found in public clouds and the level of details that can be obtained using private testbeds.

Next, we provide more details on two promising research directions that we will explore in BigFoot.

5.2.1 Performance Interference

From measurements run on Amazon EC2, Wang et. al. [106] observe that processor sharing causes unpredictable TCP and UDP throughput, varying between 1GB/s and zero for small instances. Furthermore, they've also observed abnormally large delays between end-hosts within the same datacenter, that are up to a hundred times larger than the propagation delay between the two-end hosts. One possibility for such delays could be because of queuing delays that could manifest in the drivers of the virtualization infrastructure. A comparison of different cloud hosting providers by Li et. al. [71] also show variations in network performance metrics within and across different providers.

In such virtualized hosting centers, it is possible for co-located VMs to interfere with each others' performance. For instance, recent work [91] has demonstrated the co-location of a Trajan VM with a target VM. This is used by the malicious tenant to affect the network performance of the victim VM. Seawall [97] addresses the challenge of data-center network sharing, by using hypervisor-to-hypervisor, point-to-multipoint, congestion control. With Seawall, administrators can define and enforce policies for controlling network capacity allocation between different tenants, that is agnostic to the number of flows, protocols, or destinations in the application's traffic. In BigFoot we will develop methods to infer possible interference of application that are executed in a shared environment.

5.2.2 Virtual Network Topologies

The work in [80] presents a Flat Datacenter File System, which addresses the issue of **data locality** in data-intensive applications. It is well known that, in the context of large-scale data analytics, *moving data is more expensive than moving computation*: due to the sheer size of datasets on which data analysis job operate, the network becomes a severe bottleneck and moving data to compute nodes that operate on it, may take a large fraction of the job execution time. For this reasons, several frameworks including MapReduce, strive for placing data once and for all on cluster nodes (modulo server failures and load balancing) and send computation to the nodes holding the data. As such, scheduling disciplines that allocate resources of the cluster to a particular job need to take great zeal in maintaining data locality. The main culprit of poor network performance in a datacenter is due to the hierarchical topology that is typically used to lay down the datacenter network fabric. In [80], the authors propose to use recent advances in datacenter networks such as [24, 48],and design a novel distribute filesystem that exploit full bisection bandwidth available from the network layer. They also revisit the whole networking stack by suggesting new routing and transport protocols capable of exploiting the available bandwidth of the datacenter. In conclusion, the proposed distributed filesystem simplifies the design of data-intensive application because essentially reading data from the local disk or from a disk residing in a remote machine becomes equivalent: data locality is not a fundamental issue any more.

Despite the promises of the approach outlined above, in BigFoot we have to consider an additional challenge: tenant networks are virtual, in the sense that network equipment is software-based (not hardware) and network cables are used only to transport traffic across physical machines. In this context, little is known about network performance, data locality problems and to which extent the network topologies suggested for physical datacenter fabrics would be beneficial for virtual fabrics. Our plan is to investigate, with a measurement approach, the performance of such virtual networks and study whether software switches can also benefit from redundant (virtual) topologies.

6 Conclusion

We argue that virtualization is a key enabler for the BigFoot software stack. Accordingly, this document has reviewed the state-of-the-art virtualization literature and technology that facilitates the flexible allocation of resources to solve big computational analysis tasks.

In summary, while hardware virtualization is by now a mature field, there are several important aspects which are not well understood today: especially network and, in general, I/O virtualization are largely unexplored fields. Network and disk I/O from VMs however constitute a bottleneck for data-intensive applications and new approaches to VM placement and "software network fabric" are required. While we have discussed some initial solutions in the literature, little experimental evidence about their performance exists.

Moreover, we believe that there is a big potential in taking a tenant-centric perspective, in order to improve a service. Not much is known yet on how to exploit the flexibilities introduced through virtualization, with respect to how resources can be adapted flexibly to meet certain deadlines on computation tasks or to deal with unexpected delays, failures or specifics of the application.

Our goal is to fill these gaps: first, with a measurement approach, we will set off to understand the typical workloads that hinder the operation of a datacenter, with emphasis on the network layer. Then, we will focus on resource management and migration algorithms that exploit the flexibility enabled by virtualization, with the goal of improving datacenter performance.

References

[1] Akamai and AT&T Forge Global Strategic Alliance to Provide Content Delivery Network Solutions. `http://www.akamai.com/html/about/press/releases/2012/press_120612.html`.

[2] Akamai and Orange form Content Delivery Strategic Alliance. `http://www.akamai.com/html/about/press/releases/2012/press_112012_1.html`.

[3] Akamai Facts & Figures. `http://www.akamai.com/html/about/facts_figures.html`.

[4] Amazon elastic compute cloud (ec2). `http://aws.amazon.com/ec2/`.

[5] Amazon web services: Elastic mapreduce. http://aws.amazon.com/elasticmapreduce/.

[6] Apache Hadoop. http://hadoop.apache.org/.

[7] Data center map. http://http://www.datacentermap.com.

[8] Equinix. http://www.equinix.com.

[9] EuroIX, list of IXPs. https://www.euro-ix.net/resources-list-of-ixps.

[10] Google Datacenters. http://www.google.com/about/datacenters/.

[11] GoogleCache. http://ggcadmin.google.com/ggc.

[12] Linux containers. http://lxc.sourceforge.net/.

[13] Netflix Open Connect. https://signup.netflix.com/openconnect.

[14] OpenStack – Open Source Cloud Computing Software. http://www.openstack.org/.

[15] Quantum. https://wiki.openstack.org/wiki/Quantum.

[16] Swisscom and Akamai Enter Into a Strategic Partnership. http://www.akamai.com/html/about/press/releases/2013/press_031413.html.

[17] T-Systems to offer customers VMware vCloud Datacenter Services. http://www.telekom.com/media/enterprise-solutions/129772.

[18] Uptime Institute. http://uptimeinstitute.com/,.

[19] Network Functions Virtualisation. SDN and OpenFlow World Congress, Oct 2012.

[20] 1&1 hosting. http://www.softlayer.com/about/datacenters.

[21] V. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z. L. Zhang. Unreeling Netflix: Understanding and improving multi-CDN movie delivery. In *Proc. IEEE INFOCOM*, 2012.

[22] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger. Anatomy of a Large European IXP. In *Proc. ACM SIGCOMM*, 2012.

[23] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig. Web Content Cartography. In *Proc. ACM IMC*, 2011.

[24] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, page 6374, New York, NY, USA, 2008. ACM.

[25] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: dynamic flow scheduling for data center networks. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, page 1919, Berkeley, CA, USA, 2010. USENIX Association.

[26] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H.. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. UC Berkeley Technical Report EECS-2009-28, 2009.

[27] Dushyant Arora, Anja Feldmann, Gregor Schaffrath, and Stefan Schmid. On the benefit of virtualization: Strategies for flexible server allocation. In *Proc. USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE)*, Boston, USA, March 2011.

[28] M. Axelrod. The Value of Content Distribution Networks. AfNOG 2008, `http://www.afnog.org/afnog2008/conference/talks/Google-AFNOG-presentation-pub`

[29] Microsoft Azure. `http://www.windowsazure.com`.

[30] Nikhil Bansal, Kang-Won Lee, Viswanath Nagarajan, and Murtaza Zafer. Minimum congestion mapping in a cloud. In *Proc. 30th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 267–276, 2011.

[31] A. Belbekkouche, M. Hasan, and A. Karmouch. Resource discovery and allocation in network virtualization. *IEEE Communications Surveys Tutorials*, (99):1–15, 2012.

[32] Theophilus Benson, Aditya Akella, and David A. Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, page 267280, New York, NY, USA, 2010. ACM.

[33] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. The case for fine-grained traffic engineering in data centers. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, INM/WREN'10, page 22, Berkeley, CA, USA, 2010. USENIX Association.

[34] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. Understanding data center traffic characteristics. *SIGCOMM Comput. Commun. Rev.*, 40(1):92–99, Jan 2010.

[35] Marcin Bienkowski, Anja Feldmann, Dan Jurca, Wolfgang Kellerer, Gregor Schaffrath, Stefan Schmid, and Joerg Widmer. Competitive analysis for service migration in vnets. In *Proc. ACM SIGCOMM VISA Workshop*, pages 17–24, New Delhi, India, August 2010. ACM.

[36] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, 1998.

[37] Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal online algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992.

[38] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg. Live Wide-Area Migration of Virtual Machines Including Local Persistent State. In *VEE*, 2007.

[39] Nabeel Farooq Butt, Mosharaf Chowdhury, and Raouf Boutaba. Topology-awareness and reoptimization mechanism for virtual network embedding. In *Proc. IFIP/TC6 NETWORKING*, 2010.

[40] Mosharaf Kabir Chowdhury and Raouf Boutaba. A survey of network virtualization. *Elsevier Computer Networks*, 54(5), 2010.

[41] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. An improved algorithm for matching large graphs. In *Proc. Workshop on Graph-based Representations in Pattern Recognition*, 2001.

[42] Andrew R. Curtis, Jeffrey C. Mogul, Jean Tourrilhes, Praveen Yalagandula, Puneet Sharma, and Sujata Banerjee. DevoFlow: scaling flow management for high-performance networks. *SIGCOMM Comput. Commun. Rev.*, 41(4):254265, Aug 2011.

[43] F. Dobrian, A. Awan, I. Stoica, V. Sekar, A. Ganjam, D. Joseph, J. Zhan, and H. Zhang. Understanding the Impact of Video Quality on User Engagement. In *Proc. ACM SIGCOMM*, 2011.

[44] Guy Even, Moti Medina, Gregor Schaffrath, and Stefan Schmid. Competitive and deterministic embeddings of virtual networks. In *Proc. 13th International Conference on Distributed Computing and Networking (ICDCN)*, pages 106–121, 2012.

[45] Jinliang Fan and Mostafa H. Ammar. Dynamic topology configuration in service overlay networks: A study of reconfiguration policies. In *Proc. IEEE INFOCOM*, 2006.

[46] A. Gerber and R. Doverspike. Traffic Types and Growth in Backbone Networks. In *OFC/NFOEC*, 2011.

[47] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The Cost of a Cloud: Research Problems in Data Center Networks. *ACM CCR*, 2009.

[48] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Pate, and Sudipta Sengupta. Vl2: a scalable and flexible data center network. *ACM SIGCOMM CCR*, 39:51–62, 2009.

[49] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. VL2: a scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, page 5162, New York, NY, USA, 2009. ACM.

[50] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. BCube: a high performance, server-centric network architecture for modular data centers. *SIGCOMM Comput. Commun. Rev.*, 39(4):6374, Aug 2009.

[51] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. Dcell: a scalable and fault-tolerant network structure for data centers. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, page 7586, New York, NY, USA, 2008. ACM.

[52] Aun Haider, Richard Potter, and Akihiro Nakao. Challenges in resource allocation in network virtualization. In *Proc. ITC Specialist Seminar on Network Virtualization*, 2009.

[53] Fang Hao, T. V. Lakshman, Sarit Mukherjee, and Haoyu Song. Enhancing dynamic cloud-based services using network virtualization. *SIGCOMM Comput. Commun. Rev.*, 40(1):67–74, 2010.

[54] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Mesos: A platform for fine-grained resource sharing in the data center. In *USENIX NSDI*, 2011.

[55] Ballani Hitesh, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards predictable datacenter networks. In *ACM SIGCOMM*, 2011.

[56] U. Hoelzle and L. A. Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machine*. Morgan and Claypool Publishers, 2009.

[57] I. Houidi, W. Louati, and D. Zeghlache. A distributed virtual network mapping algorithm. In *Proc. IEEE ICC*, 2008.

[58] C. Huang, A. Wang, J. Li, and K. Ross. Measuring and Evaluating Large-scale CDNs. In *Proc. ACM IMC*, 2008.

[59] Sandvine Inc. Global broadband phenomena. Research Report `http://www.sandvine.com/news/global_broadband_trends.asp`.

[60] Virajith Jalaparti, Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Bridging the tenant-provider gap in cloud services. In *ACM SOCC*, 2012.

[61] S. Kandula, J. Padhye, and P. Bahl. Flyways to de-congest data center networks. New York City, NY, USA, 2009.

[62] Srikanth Kandula, Sudipta Sengupta, Albert Greenberg, Parveen Patel, and Ronnie Chaiken. The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, page 202208, New York, NY, USA, 2009. ACM.

[63] M. Korupolu, A. Singh, and B. Bamba. Coupled Placement in Modern Data Centers. In *Proc. IEEE IPDPS*, 2009.

[64] R. Krishnan, H. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving Beyond End-to-end Path Information to Optimize CDN Performance. In *Proc. ACM IMC*, 2009.

[65] Linux Kernel Virtual Machine (KVM). http://www.linux-kvm.org.

[66] C. Labovitz, S. Lekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet Inter-Domain Traffic. In *Proc. ACM SIGCOMM*, 2010.

[67] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros. Distributed Placement of Service Facilities in Large-Scale Networks. In *Proc. IEEE INFOCOM*, 2007.

[68] Nikolaos Laoutaris, Georgios Smaragdakis, Pablo Rodriguez, and Ravi Sundaram. Delay Tolerant Bulk Data Transfers on the Internet. In *Proc. ACM SIGMETRICS*, 2009.

[69] Leaseweb hosting. http://www.leaseweb.com/en/about-us/our-company.

[70] T. Leighton. Improving Performance on the Internet. *Communications of ACM*, 2009.

[71] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. CloudCmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, IMC '10, page 114, New York, NY, USA, 2010. ACM.

[72] Jens Lischka and Holger Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proc. ACM SIGCOMM VISA*, 2009.

[73] J. Lu and J. Turner. Efficient mapping of virtual networks onto a shared substrate. In *WUCSE-2006-35, Washington University*, 2006.

[74] H. Madhyastha, J. C. McCullough, G. Porter, R. Kapoor, S. Savage, A. C. Snoeren, and A. Vahdat. scc: Cluster Storage Provisioning Informed by Application Characteristics and SLAs. In *Proc. FAST*, 2012.

[75] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38:69–74, Mar 2008.

[76] Rich Miller. Estimate: Amazon cloud backed by 450,000 servers. http://www.datacenterknowledge.com/archives/2012/03/14/estimate-amazon-cloud-

[77] Jayaram Mudigonda, Praveen Yalagandula, Mohammad Al-Fares, and Jeffrey C. Mogul. SPAIN: COTS data-center ethernet for multipathing over arbitrary topologies. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, page 1818, Berkeley, CA, USA, 2010. USENIX Association.

[78] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. *PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric.*

[79] Limelight Networks. `http://www.limelightnetworks.com/platform/cdn`.

[80] Edmund B. Nightingale, Jeremy Elson, Jinliang Fan, Owen Hofmann, Jon Howell, and Yutaka Suzue. Flat datacenter storage. In *ACM OSDI*, 2012.

[81] E. Nygren, R. K. Sitaraman, and J. Sun. The Akamai Network: A Platform for High-performance Internet Applications. *SIGOPS Oper. Syst. Rev.*, 2010.

[82] OVH. `http://www.ovh.co.uk/aboutus/`.

[83] I. Poese, B. Frank, B. Ager, G. Smaragdakis, and A. Feldmann. Improving Content Delivery using Provider-Aided Distance Information. In *Proc. ACM IMC*, 2010.

[84] I. Poese, B. Frank, S. Knight, N. Semmler, and G. Smaragdakis. PaDIS Emulator: An Emulator to Evaluate CDN-ISP Collaboration. In *SIGCOMM demo*, 2012.

[85] I. Poese, B. Frank, G. Smaragdakis, S. Uhlig, A. Feldmann, and B. Maggs. Enabling Content-aware Traffic Engineering. *ACM CCR*, 42(5):21–28, October 2012.

[86] Lucian Popa, Gautam Kumar, Mosharaf Chowdhury, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. Faircloud: Sharing the network in cloud computing. In *ACM SIGCOMM*, 2012.

[87] The Xen Project. `http://www.xen.org`.

[88] QEMU. `http://www.qemu.org`.

[89] Rackspace. `http://www.rackspace.com/information/aboutus/`.

[90] Robert Ricci, Chris Alfeld, and Jay Lepreau. A solver for the network testbed mapping problem. *ACM SIGCOMM CCR*, 33(2), 2003.

[91] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, page 199212, New York, NY, USA, 2009. ACM.

[92] Luigi Rizzo and Giuseppe Lettieri. Vale, a virtual local ethernet. In *ACM SIGCOMM Conext*, 2012.

[93] L. Schubert, K. Jeffery, and B. Neidecker-Lutz. A Roadmap for Advanced Cloud Technologies under H2020. Recommendations by the Cloud Expert Grop.

[94] L. Schubert, K. Jeffery, and B. Neidecker-Lutz. The Future of Cloud Computing: Opportunities for European Cloud Computing Beyond 2010. EU Expert Group Report, public version 1.0.

[95] AMAZON Web Services. `http://aws.amazon.com`.

[96] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar. Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service. In *Proc. ACM SIGCOMM*, 2012.

[97] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha. Sharing the data center network. In *Proc. USENIX/ACM NSDI*, 2011.

[98] Alan Shieh, Srikanth Kandula, Albert Greenberg, Changhoon Kim, and Bikas Saha. Sharing the data center network. In *USENIX NSDI*, 2011.

[99] Softlayer. `http://www.softlayer.com/about/datacenters`.

[100] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar. Answering What-if Deployment and Configuration Questions with Wise. In *Proc. ACM SIGCOMM*, 2009.

[101] S. Triukose, Z. Wen, and M. Rabinovich. Measuring a Commercial Content Delivery Network. In *Proc. WWW*, 2011.

[102] Oracle Virtualbox. `https://www.virtualbox.org`.

[103] C. Waldspurger and M. Rosenblum. I/O virtualization. *Communications of ACM*, 55(1), 2012.

[104] G. Wang, D.G. Andersen, M. Kaminsky, M. Kozuch, TS Ng, K. Papagiannaki, M. Glick, and L. Mummert. Your data center is a router: The case for reconfigurable optical circuit switched paths. In *ACM HotNets VIII*, page 62, New York City, NY, USA, 2009.

[105] Guohui Wang and Eugene Ng. The impact of virtualization on network performance of amazon ec2 data center. In *IEEE INFOCOM*, 2010.

[106] Guohui Wang and T. S. Eugene Ng. The impact of virtualization on network performance of amazon EC2 data center. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, page 11631171, Piscataway, NJ, USA, 2010. IEEE Press.

[107] Richard Wang, Dana Butnariu, and Jennifer Rexford. OpenFlow-based server load balancing gone wild. In *Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*, Hot-ICE'11, page 1212, Berkeley, CA, USA, 2011. USENIX Association.

[108] J. Whiteaker, F. Schneider, and R. Teixeira. Explaining packet delays under virtualization. *ACM SIGCOMM CCR*, 2011.

[109] Christo Wilson, Hitesh Ballani, Thomas Karagiannis, and Ant Rowtron. Better never than late: meeting deadlines in datacenter networks. In *Proc. ACM SIGCOMM Conference*, pages 50–61, 2011.

[110] Yufeng Xin, Ilia Baldine, Anirban Mandal, Chris Heermann, Jeff Chase, and Aydan Yumerefendi. Embedding virtual topologies in networked clouds. In *Proc. 6th International Conference on Future Internet Technologies (CFI)*, pages 26–29, 2011.

[111] Mei Yiduo, Ling Liu, Xing Pu, and Sankaran Sivathanu. Performance measurements and analysis of network i/o applications in virtualized cloud. In *IEEE CLOUD*, 2010.

[112] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM CCR*, 38(2), 2008.

[113] Y. Zhu and M. Ammar. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. In *Proc. IEEE INFOCOM*, 2006.