Towards a Method Integrating Virtual Switch Performance Into Data Centre Design

Mitalee Sarker* and ‡, Jan Siersch†, Arslan Khan* and Stefan Wesner* and †
*Institute of Information Resource Management
Ulm University, Germany
Email: [firstname.lastname]@uni-ulm.de
†Kommunikations und Informationszentrum (kiz)
Ulm University, Germany
Email: stefan.wesner@uni-ulm.de
‡ Landeshochschulnetz BelWü, Germany
Email: mitalee.sarker@belwue.de

***Abstract*—Data Centre Design is a complex task due to the wide range of technological options and building blocks available. Deriving the data centre system architecture is done based on a couple of uncertain assumptions rather than known facts. The *future* workload of users or, more precisely, of their applications can only be predicted. Furthermore, the benchmarking and design process is typically driven by hardware features and options. In this paper, we argue that an important aspect to be considered in the design process is not only the network capabilities and topology but also the intended virtual switch solutions and their performance potentially jeopardising the overall system quality. The paper contains preliminary work based on a new network-aware algorithm for virtual machine placement in a virtualised cloud environment. Our initial evaluations help to create a fast and effective decision-making model for such an algorithm.**

Keywords–Data Centre Design; Virtual Switch; Software Defined Networking Performance.

I. INTRODUCTION

The design of a data centre is inherently complex and covers a wide range of technical disciplines. The task starts with considerations on the server capabilities, their specific configurations in terms of *Central Processing Unit* (CPU) types, frequency, number of cores, as well as all the other hardware components such as main memory, I/O system, network connectivity and network topology. At the end of the process, the facility infrastructure planning covers power supply and distribution, cooling approaches and optimisation, as well as specific floor plans describing the layout of the racks in rows within the computing room, and preparing the set-up and operation of the system. The latter part has gained particular attention in the last years as values for achieving power efficiency e.g., expressed with values, such as the Power Use Efficiency (PUE), have become an essential part of the design process.

Some data centre systems are designed for highly resource demanding applications such as for technical simulations or big data analytics. Before a procurement or buying decision for those data centres is done, it is common to realise a tailored data centre design. This design aims to deliver a system with a balanced server architecture by avoiding a system where one or a combination of several components limits the overall system performance. For example, as discussed in [1], this process requires an in-depth analysis of collected usage and performance data of a previous or similar systems, and varies significantly for different user groups and disciplines, even within a single application domain. Therefore, it is common to have no homogeneous system architecture, but a combined and jointly operated system with several segments that might differ significantly in their server architecture [2].

Many Cloud computing data centres are designed to cover a wide range of applications. They are optimised for low costs that can be best achieved with a system architecture that is based on simple components and is as homogeneous as possible. In this paper though, we focus on *heterogeneous* Cloud data centres that have optimised servers for hosting different types of *Virtual Machines* (VMs), and perform an active management not only for the initial placement of VMs but also for their migration to different servers based on observed performance, as proposed in [3]. This is particularly true if the targeted applications are resource demanding, such as *High Performance Computing* (HPC) simulations or High Performance Data Analytics (HPDA) [4] workloads. In traditional HPC data centres, the analysis and system design can be focused on the application workload and the operating system [5]. For a Cloud environment, the overhead of the hypervisors and, as we argue here, the network infrastructure play a key role and must be considered jointly with the hardware options. In other words, selecting the hardware system independently from the targeted virtualisation infrastructure can lead to severe bottlenecks and underutilisation.

We claim that, due to the currently observed performance characteristics of virtual switches, which are included in cloud environments such as OpenStack, have a severe impact on the overall data centre design. In particular, for Cloud data centres with demanding applications, the analysis of limiting capabilities of the underlying physical infrastructure is a common approach for selecting the most appropriate hardware for a given set of applications. This is commonly done by defining a set of benchmark applications that represents a typical workload in all of its aspects, from CPU load, over memory and external storage access patterns down to its communication behaviour. In general, these application-driven benchmarks are complemented by synthetic benchmarks. In order to provide a method to deliver comparable results across different system configurations that are considered as technological basis for a new data centre or, to find the most appropriate hosts within a given data centre, these synthetic benchmarks represent isolated elements with well understood and repeatable behaviour.

In a virtualised environment, in principle, the same approach can be followed with the extension that the analysis of

 ISBN: 978-1-61208-450-3

the physical hardware is no longer enough, but a set of virtual machines operating at the same time on a host in a defined mix has to be considered. Due to well advanced capabilities in hypervisors to virtualise CPU and memory resources, the loss of performance compared to a non-virtualised operation has decreased visibly with newer hardware generations, and has provided the foundation to include highly demanding applications within a virtualised environment.

The quality of the network connectivity plays a major role not only for the communication with other VMs, but also for accessing remote storage systems. Despite its key role in delivering overall performance, quite surprisingly, the network virtualisation is still often done with a purely software-based approach rather than relying on hardware level features.

In this paper, we introduce an initial model on how the performance limitations of virtual switch solutions can have an impact on the configuration options for a Cloud system architecture. In particular, these limitations represent a major caveat for demanding applications, and make the adoption of the Cloud model for such high performance applications less attractive.

A. Problem Statement

More specifically, the questions that need to be tackled in order to address the aforementioned issues are:

1) How could existing benchmarking and design approaches from HPC Data Centre design be extended or re-used to determine network bottlenecks in Cloud data centres?
2) Which methods can be used to identify performance properties of virtual switches and derive a model to predict the performance for different load situations?
3) How can such a model be used to validate the suitability of potential system architectures?

B. Related work

The work done in [6] is focused on VM placement and traffic routing in the data centre network in order to reduce the traffic cost. One online algorithm was introduced for the dynamic arrival and departure of VMs. Another online algorithm based on Markov approximation method was also presented by the authors, which performs a tradeoff between performance and cost. Although their work showed significant improvement on very large and small flows of data centre network, they have not considered the impact of the virtual switch on their proposed algorithms. In [7] Cohen et. al. discuss options for bandwidth-constrained VM placement optimisation problems. The algorithm is focused on storage area networks and is depicted from the communication between VMs and the core of the data centre network that connects the storage devices to the data centre network. Neither of these approaches has taken virtual switch constraints into account.

C. Solution Approach

The assumptions made for our approach are that the data centre considered is not a general purpose system, potentially hosting *all* kinds of applications with a completely *unpredictable* behavior, but certain knowledge is available about the types of applications hosted and their behavior. Based on these assumptions, application benchmarks creating a realistic and representative load can be derived, and synthetic benchmarks can be created by putting artificial load e.g., on the network to perform tests in a repeatable and consistent manner across different systems.

The remainder of this paper is structured as follows: Section II provides a theoretical model for the VM placement decision in a virtual switch aware Cloud data centre. Section III discusses our work on evaluating the performance characteristics of current virtual switch solutions for the network traffic model. Section IV presents the results gathered from these performance measurements, and interprets them. Lastly, Section V contains the conclusions drawn from this paper and an outlook on future work.

II. Network unaware VM placement

The challenge for an initial placement of VMs can be described as finding a distribution of k VMs $(v_1, v_2, \ldots, v_k)$ with a known maximum resource demand that is defined by a metric such as number of virtual cores, memory or disk space on a number of n servers with potentially different capabilities and load situation $S_1, S_2, \ldots, S_n$. Current models often focus on memory and compute core demand. If shared storage is used, no disk bottlenecks need to be considered with respect to capacity. A simplified algorithmic view assuming core and memory use as driving metrics is shown in the algorithm in Figure 1.

```
function SimpleCount(vmList)

    coresConsumed = 0
    memoryConsumed = 0
    vmCount = 0
    while vmList.current()≠ null do
        memoryConsumed += vmList[i].memoryReq
        coresConsumed += vmList[i].coreReq
        if currentMemory ≤ memMax
&& currentCoresUsed ≤ coreMax then
            vmCount +=1
            vmList.next()
        else
            // we exceeded the available resources...
            exit
        end if
    end while
    return vmCount
end function
```

Figure 1. Simple network agnostic VM packing algorithm

This almost naive algorithm is not uncommon in current Cloud implementations and sometimes only memory footprint of a VM is considered [8]. While it is definitely not a very good approximation of changing workload demands, it is commonly used as it maps nicely on the instance types. Instance types are the units that represent how resources within a cloud are typically offered. In addition, these metrics are well understood, as they relate to the typical selection criteria of physical servers in the past. Furthermore, for many applications in the data centre, the available network capacity of several 10Gbps, 40Gbps or even 100Gbps ports exceed the typical demand visibly. In order to address concerns related to security and potential influence

 ISBN: 978-1-61208-450-3

on performance on these shared network resources, virtual switches have been established as a common approach to act similarly to the hypervisor for CPU and memory resources as gatekeeper between different VMs.

We consider in this paper, in particular resource demanding applications in terms of communication. For example, due to significant I/O traffic on a network provided storage device, not only the performance of the network links but also the performance of the Virtual Switch becomes a concern.

A. Variety of virtual switch approaches

There are several virtual switch products currently available and they differ fundamentally in the way they are implemented. This section aims to provide a compact overview on the different switching approaches. In general, these switches consist of a data plane and a control plane. The data plane handles packet manipulation and contains the forwarding logic. The control plane manages the rules under which the switch operates through some sort of management protocol. In terms of implementation, there exists a spectrum from fully software-based switching approaches to integration of switching logic directly into hardware components. The detailed performance implications of these design decisions have yet to be evaluated.

Open vSwitch (OVS) [9] is an example of a purely software-based managed virtual switch [10], [11]. It is often used for network virtualisation in cloud environments such as OpenStack. Open vSwitch supports a variety of network virtualisation and management protocols. Fundamentally, the switch consists of a kernel space data plane and user space control plane. To avoid costly context switches between the kernel and user space components, flow caching is implemented as part of the kernel side logic of the switch.

The *Intel Data Plane Development Kit* (DPDK) [12] is a collection of user space application libraries dedicated to high-performance packet processing [13]. It improves performance by providing applications with multi-core enabled data plane functionality and poll mode Network Interface Card (NIC) drivers, which operate directly in user space. These drivers are available for 1GbE and 10GbE interfaces. *Intel DPDK vSwitch* is based on Open vSwitch, and modified to take advantage of the functionality provided by DPDK. This increases its packet switching performance, especially when handling large quantities of small packages [14], [15].

Lagopus [16] is a software-based, OpenFlow 1.3 compliant SDN switch with support for network function virtualisation and multiple data plane implementations [17]. Communication between the data plane and the switch agent takes place through an event queue mechanism. There exist multiple data plane implementations for the switch, such as raw sockets, Intel DPDK, and bare-metal switch variants that benefit from hardware acceleration.

In contrast to the previously mentioned products, the *Mellanox eSwitch* takes advantage of switching capabilities embedded in supported NICs [18], [19]. A user space daemon provides access to the switch's functionality. Moving the data plane directly into hardware yields the potential of improved performance compared to purely software-based solutions. In addition, it supports *single root I/O virtualisation* (SR-IOV) to multiplex multiple virtual interfaces (virtual functions) into a single physical interface (physical function). This, in combination with the embedded switching capabilities, removes the need for software-based virtual network devices for the connection of guest instances. This in turn leads to improved performance, while simultaneously reducing the CPU load of the physical host. In such a scenario, the eSwitch can handle packet flows between the physical and virtual functions, apply packet filtering rules (including protection against *Media Access Control* (MAC) spoofing), and isolate virtual networks using *Virtual Local Area Networks* (VLANs). However, in order to use eSwitch, specialised *Network Interface Cards* (NICs) are required, and the number of virtual functions available to the host is limited by the internal architecture of the NIC.

B. A virtual switch aware design method

As outlined in the previous section, different approaches exist to realise virtual switches. From purely software based over NIC aware up to NIC embedded switch solutions, one can also expect a wide range of different performance characteristics. As a result, the network demand, along with the current common metrics, needs to be included in the decision in order to make a suitable initial placement of VMs and perform migration and optimisation steps during operation.

Compute and memory demand can be considered as separate due to the capabilities of the underlying hypervisor. Although virtualisation can be achieved by using the virtual switches, the traffic produced by the different VMs is combined and is using the shared network resources per server. As a result, a more complex model for modelling the network demand is necessary where the overlay of the traffic from all VMs within a server needs to be considered.

Initially, consider the following model for a VM's network load as shown in Equation (1). It is very simple and it assumes that with a probability p, the VM_i is sending with a fixed rate of k kbps, and with probability $1-p$, no traffic is generated.

$$\mathrm{VM}_i = \begin{cases} k\ kbps & p \\ 0\ kbps & 1-p \end{cases} \tag{1}$$

A simple traffic model can be achieved from this initial model by using values for average bandwidth k and probability p for active and inactive periods, as well as an average time in state *active* and *inactive* based on traffic analysis. This traffic model can be used in a simulation to estimate average workload of different VMs combined at a virtual switch. It can be further extended by defining several states with different bandwidth demands and a matrix expressing the probabilities to move from a state i to a state j. With this model, we have assumed for the sake of simplicity, Markov properties that only the previous state is relevant. Furthermore, we can now define a matrix of probabilities Q for changing from state i to j as shown in Equation (2).

$$Q = \begin{pmatrix} 0 & q_{01} & \cdots & q_{0(n-1} \\ q_{10} & 0 & \cdots & q_{1(n-1)} \\ \vdots & \cdots & \ddots & \vdots \\ q_{(n-1)0} & \cdots & q_{(n-1)(n-2)} & 0 \end{pmatrix} \tag{2}$$

 ISBN: 978-1-61208-450-3

Together with the probability or time to stay in a state, we have a more fine grained model for the traffic that can be expected from a given VM_i. These probabilities can be either derived from the nature of the VM performing a white box modelling or, by observing the properties using traffic analysis tools and discretising the measured bandwidth to a defined number of different states and their changed probabilities.

In order to make a prediction, if a given server using a specific virtual switch is overloaded with a load of n VMs, one also needs to model the capabilities of the virtual switch as well. Again, a mix of the white box approach e.g., derive the overhead of certain methods to realise the tunnels, and black box modelling to determine how the outgoing bandwidth changes based on input load, has to be used. An initial model could be using the simple VM model with on and off states as a prediction, for how many VMs, a virtual switch can deliver the accumulated bandwidth as outgoing traffic. So the initial virtual switch model would correspond to the sum depicted in Equation (3).

$$\mathrm{BW}_{out} = \min\left\{\sum_{1}^{n} \mathrm{VM}_i \; ; \text{saturation bw}\right\} \qquad (3)$$

While this initial simple model for the switch is clearly less complex than the ones you can find in network simulators, it can be used to derive very quick decisions within a VM placement algorithm. For example, we can use it to decide if a given VM fits within a given server or, how many VMs can be hosted by a given server architecture.

Based on this initial and simple models, it is necessary to derive from the VMs, the probabilities for changing state and the time how long they stay in a given state, as well as the saturation bandwidth of the virtual switch on a given server system.

III. Realisation Approach

In this section, we have summarised our initial results, in order to determine the limits of different virtual switch configurations and collect initial data on the limits for given server systems. Furthermore, we have used tools that can collect traces from VMs on their communication behaviour in order to develop a traffic model for them.

A. Performance overhead introduced by Open vSwitch VXLAN encapsulation

For the purpose of measuring the performance overhead introduced by network virtualisation, and to establish a plausible baseline for the possible aggregate network throughput of a server equipped with a software-based virtual switch, a test setup was created where both physical non-virtualised networks and virtualised networks using OVS managed *Virtual eXtensible Local Area Network* (VXLAN) tunnels were available. Both network types utilised the same underlying networking hardware, interface configuration and software components, aside from the addition of the tunnelling mechanism itself. Network performance measurements were then conducted from inside virtual machines (instances), which were created and managed through OpenStack [20]. The instances were equipped with two separate tap devices, one connected to the physical network and the other connected to a virtualised network which was transported over the same physical interface.

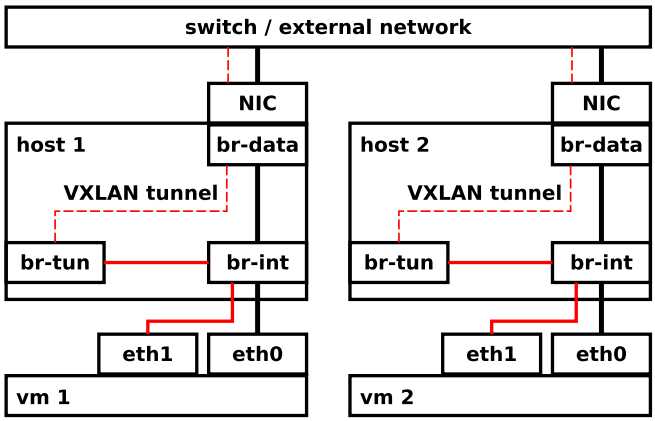

Figure 2. Simplified schematic of network components used for Open vSwitch tunnelling in OpenStack

The OpenStack installation itself consisted of dedicated controller VMs and several physical compute nodes, two of which were used exclusively for the measurements. Each host contained one measurement instance. The physical networking was based on Mellanox ConnectX3-Pro single-port 56GbE QSFP NICs connected to a Mellanox SX1012 12-port 56GbE QSFP switch. The NICs were operated in 56GbE mode with the default *Maximum Transmission Unit* (MTU) of 1500 bytes. Aside from the NICs and the switch, no other physical networking device was located inside the network path for the performance measurements. A more detailed list of specifications for the physical and virtual resources can be found in Appendix A.

In the OpenStack Neutron (networking service) configuration with OVS, which was outlined in Figure 2, the flat external network (solid line from the switch down to eth0) was mapped to an OVS bridge called *br-data*. This bridge was attached to the physical NIC (similar to the external bridge br-ex for network nodes in default OpenStack installations). The virtual tenant networks were attached to the OVS tunnelling bridge *br-tun*, where packets were encapsulated using VXLAN (dotted line) before being transported over the physical NIC. The integration bridge *br-int* connected the various other virtual network components and host-locally isolated different networks using VLANs.

Inside the virtual machines, the network devices for the flat external network and the VXLAN virtual private network were represented as *eth0* and *eth1* respectively. Both devices were configured with a MTU of 1450 bytes to avoid fragmentation on the virtualised private network. Otherwise, fragmentation could occur due to the introduction of additional header bytes by the encapsulation. This MTU configuration also kept the throughput measurements comparable between the two interfaces. To conduct the interface benchmark, the tool IPerf [21] v3.0.11 was used with 4 parallel TCP streams between the client instance and the server instance. Throughput measurements were performed for 300 seconds, to average out slow-start effects and random spikes in the TCP connections.

 ISBN: 978-1-61208-450-3

B. Performance overhead introduced by Open vSwitch VXLAN and GRE encapsulation

The approach mentioned in this section, was to compare the tunnelling protocols for bridging Cloud data centres which are located at different places. A conceptual study on tunnelling protocols was performed and from the study, it was found that VXLAN and *Generic Routing Encapsulation* (GRE) are well suited and both can be generated by using Open vSwitch.

The advantages of using VXLAN is that, it provides 16 million VXLAN ID which overcomes the ID limitation of VLAN. By using VXLAN, we can deploy overlay networks in the virtualised data centres with multi-tenant environment and layer 2 network connection through multiple data centres. The *User Datagram Protocol* (UDP) encapsulation inside VXLAN allows each *Local Area Network* (LAN) segment to be extended across layer 3. VXLAN is also cost-effective as it works in virtualised network, and it saves time as we do not need to deploy many hardware devices. However, VXLAN does not have any control plane. So the network does the work of the control plane such as allocating segment ID and multicast. As VXLAN has no significant support for security, additional measures such as adding *Internet Protocol Security* (IPSec) tunnels are necessary.

While GRE tunnels are less complex to configure and are capable of transmitting multicast traffic, the encapsulation on top of TCP/IP come with the known issues of the limited window size for high speed links. By using GRE tunnels, we can easily transmit packets containing incompatible protocols over the intermediary network via encapsulation with compatible protocols. For example, we can transfer *Internet Protocol version 4* (IPv4) packets over a network that only allows *Internet Protocol version 6* (IPv6) packets. But GRE is even less secured than VXLAN and is more complex in the tunnel set-up procedure.

A synthetic and a database use case were considered to evaluate the performance of the tunnelling protocols and the linked software components in the virtual switch testbed. The measurements included the impact of the tunnelling protocols on point-to-point throughput and latency. As synthetic use cases, an IPerf tool-based evaluation and a *File Transfer Protocol* (FTP)-based evaluation were performed to measure throughput and latency. In the FTP-based evaluation, files of different sizes were downloaded from a FTP server to a FTP client through both tunnelling protocols and the downloaded time was captured. In the database use case, Couchbase database was used to measure the network load when a new instance was added to a cluster.

The tests were performed in two different testbeds. The first testbed included two personal computers and a switch with 1 GbE NIC. The second testbed combined two MicroServers with 10 GbE NIC each. Figure 3 depicts the network topology used in both testbeds. Open vSwitch was used to create the VXLAN and GRE tunnels. In both testbeds, the network settings were kept equal to allow comparison of the results. In testbed 1, the MTU of the 1GbE physical NIC was set to 3000, and 9000 for the 10GbE NIC in testbed 2. The detailed specifications for the two testbeds can be found in Appendix B.

To simulate the distance between the Cloud data centres, network latency between the end systems was increased by using Linux kernel mechanisms in both testbeds. An additional delay between 100ms and 20ms was introduced to the network.

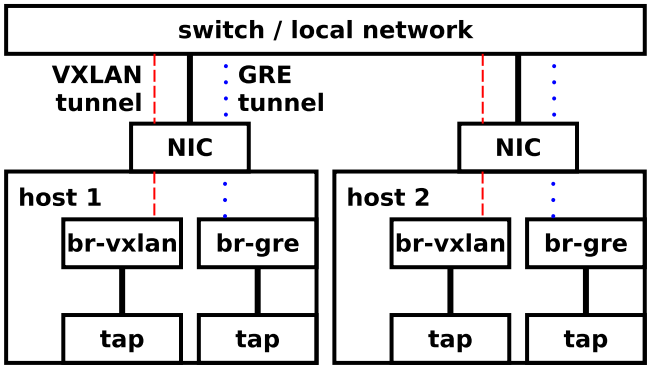

Figure 3. Simplified schematic of network components used for evaluating VXLAN and GRE tunnels in two testbeds.

IV. Results

A. Performance overhead introduced by Open vSwitch VXLAN encapsulation

The results from the measurements using the setup detailed in Section III-A show a significant drop in bandwidth when OVS VXLAN encapsulation was introduced as a network virtualisation mechanism. As seen in Figure 4, throughput in the flat external network averaged $11.289 Gbps$, whereas throughput in the VXLAN-based virtual tenant network only averaged $1.241 Gbps$, roughly one-tenth of the bandwidth in the flat network. Though not detailed in the measurements, the limiting factor in the virtualised network appears to be CPU-bound by the performance of the thread responsible for the packet encapsulation.

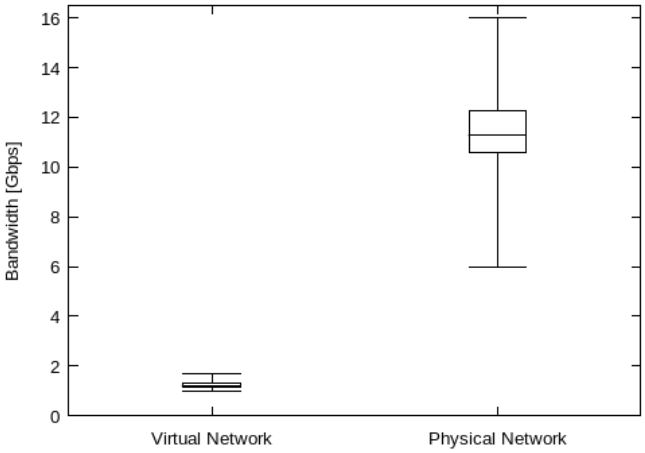

Figure 4. IPerf throughput for VXLAN-based virtual networks and flat physical networks using OpenStack with Neutron OVS networking.

In conclusion, the performance impact of OVS VXLAN encapsulation creates a serious practical limitation for its use in tenant network virtualisation with network interfaces exceeding the 1GbE standard, both in terms of raw throughput and CPU overhead. Therefore, it's use needs to be considered

 ISBN: 978-1-61208-450-3

carefully when instances with network intensive applications are expected.

B. Performance overhead introduced by Open vSwitch VXLAN and GRE encapsulation

Figure 5 represents the IPerf tool-based evaluation and contains a comparison between the bandwidth utilisation of VXLAN and GRE tunnel enabled network.

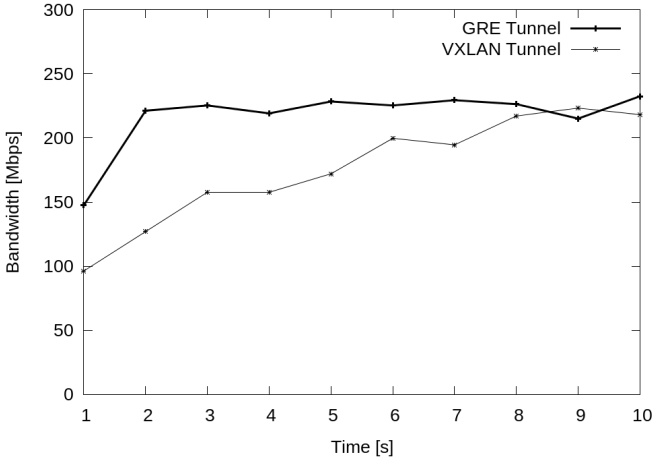

Figure 5. IPerf TCP bandwidth results with 1 stream while downloading a 3GB file through VXLAN and GRE tunnels in testbed 1.

One stream was generated while downloading a 3GB file from the server using FTP. The TCP window size for the testbed was set to the default $85.0KByte$. The measured bandwidths in the VXLAN and GRE tunnel enabled networks were $180465.00Kbits/s$ and $221307.00Kbits/s$ respectively. From the figure, it is seen that the bandwidth utilisation was decreased in both networks due to the additional network delay. But the performance of both tunnels in the network can be seen more clearly. The measured bandwidth was higher in the GRE tunnel.

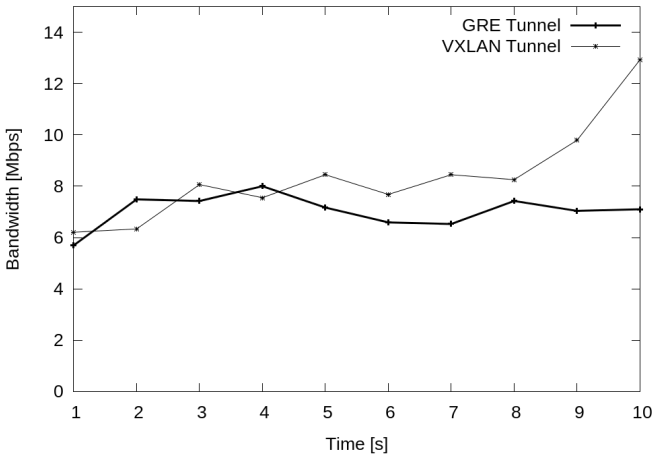

Figure 6. IPerf TCP bandwidth results with 1 stream while downloading a 3GB file through VXLAN and GRE tunnels in testbed 2.

Figure 6, taken from an IPerf tool-based evaluation in the network with 10GbE NIC, represents similar comparison mentioned in Figure 5. The TCP window size in the testbed was kept at the default of $24.5KByte$. The bandwidth utilisation in the VXLAN tunnel enabled network was $8516.00Kbits/s$, which was less than the corresponding value in the network with 1GbE NIC. Similar degradation in bandwidth is visible in the GRE tunnel enabled network. The measured bandwidth in this network was $7159.00Kbits/s$. Both results give clear indication that the network with higher bandwidth was not fully utilised by the tunnels. Also, there are larger fluctuations in the current graph compared to the graph in Figure 5.

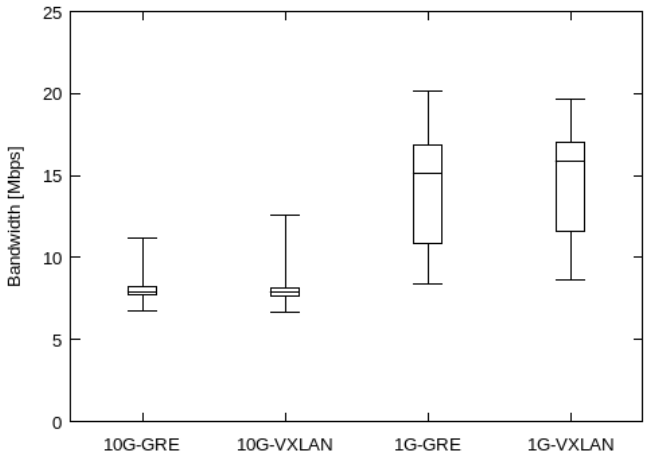

Figure 7. Performance comparison of VXLAN and GRE tunnels while downloading a 10MB file using FTP.

Figures 7 and 8 represent FTP-based evaluations in testbeds 1 and 2. The graphs indicate the impact of VXLAN and GRE tunnels on small file (10MB) and large file (5GB) downloads in both testbeds. Each measurement was repeated 50 times to average out random fluctuations. From the raw data of download time, the bandwidth usage was calculated. The X-Axis represents the testbeds with VXLAN and GRE tunnels. The Y-axis represents the boxplot of the bandwidth usage. From the bottom to the top, the boxplot shows the minimum, the first quartile, the second quartile which is the median, the third quartile and the maximum usage of bandwidth. In Figure 7, we can see that the maximum and minimum bandwidth usage for VXLAN tunnel in the 1st testbed were $19.625Mbps$ and $8.615Mbps$. For GRE tunnel, the values were $20.178Mbps$ and $8.407Mbps$. For the second testbed, the highest and lowest bandwidth usage for VXLAN tunnel were $12.558Mbps$ and $6.674Mpbs$. The corresponding values for GRE tunnel were $11.201Mbps$ and $6.712Mbps$. In comparison, both tunnels showed better performance in testbed 1 than in testbed 2 with respect to the bandwidth.

The effect of the tunnels when downloading large files in both testbeds is depicted in Figure 8. The bandwidth utilisation was higher in the 1GbE NIC network compared to the network with 10GbE NIC. The maximum bandwidth for VXLAN and GRE tunnel in testbed 1 were $36.631Mbps$ and $37.387Mbps$ respectively, whereas for testbed 2, the values were $22.013Mbps$ and $21.969Mbps$. The values for the minimum bandwidth usage followed the same pattern.

For the network with 1GbE NIC, the average performance of the VXLAN tunnel was better than the GRE tunnel. The

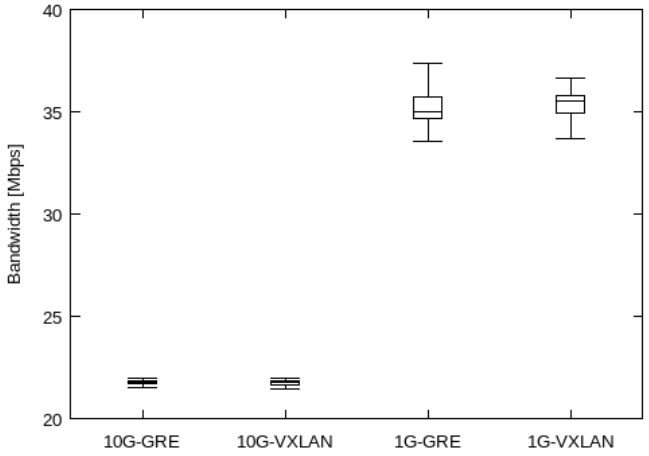

Figure 8. Performance comparison of VXLAN and GRE tunnels while downloading a 5GB file using FTP.

reason for this is the difference in the underlying transport protocols. While VXLAN is based on UDP, the GRE encapsulation operates on top of TCP. For the network with 10GbE NIC, it was detected that Open vSwitch and the weak CPUs of the MicroServers were not compatible with high workloads. Both tunnels delivered degraded performance when there was huge traffic in the network. Also, a high rise in CPU load on the physical machines and an average packet loss of more than 14% were observed during the test in that network. Network delay increased the drop rates and CPU load. Preliminary results indicate that this behaviour might be caused by a non-optimal implementation of Open vSwitch and by the consequences of using virtualisation as it disables hardware features such as TCP offloading. The investigation of this behaviour is still ongoing.

C. Impact on Data Centre Design

As shown in the previous section, depending on the chosen virtual switch solution, the number of VMs that can be placed on a specific server is not determined by CPU cores or memory constraints, but by the maximum throughput that can be achieved with the virtual switch. As the switch performance might be much less than the physical connectivity, even for less communication intensive VMs, the virtual switch performance is the limiting factor.

This leads to the result that, in order to avoid underutilised memory or compute resources, the server design must consider also the virtual switch performance. Furthermore, the chosen approach to place VMs of a customer across the data centre infrastructure has an impact on the performance that can be realised. For Open vSwitch, only small servers (low memory, low core count) with a couple of 1GbE links or, at best a single 10GbE link are cost-effective.

As outlined above, the VM placement algorithms of current Cloud middleware solutions, such as OpenStack, do not consider the communication behaviour of VMs or, constraints introduced by a specific choice of a virtual switch. Therefore, the switch characteristics must be integrated into the placement algorithms e.g., by adding custom filters and removing servers from the list of potential candidates. Figure 9 shows a basic extension that demonstrates the concept of how the placement algorithms could be amended. Still, the basic algorithms only consider average and peak values if the bandwidth demands can be supported by the deployed virtual switch. The next check is validating if the maximum number of tunnels or accumulated bandwidth has been reached.

```
    ...
    while vmList.current()≠ null do
        // check if the bandwidth demand is supported at all
    by the switch type of this server
        if  vmList[i].peakBW ≤ SwitchType.maxTunnel then
 5:         bwSum += vmList[i].averageBW;
            // check if the BW still fits into max BW and max.
    number of tunnels
            if  bwSum > SwitchType.maxPeak
    || vmCount ≥ SwitchType.maxTunnel then
                bwSum -= vmList[i].averageBW;
10:             exit
            end if
        else
            exit
        end if
15:     memoryConsumed += vmList[i].memoryReq;
        coresConsumed += vmList[i].coreReq;
        if currentMemory ≤ memMax
    && currentCoresUsed ≤ coreMax then
            vmCount +=1
20:         vmList.next()
        else
            // we exceeded the available resources...
            exit
        end if
25: end while
    return vmCount
    ...
```

Figure 9. An initial network aware placement decision algorithm.

A more realistic model would work with a switch model that would not only require average and maximum bandwidth demands but would also need the probabilities for certain

bandwidth states and would overlay them. If an additional VM would exceed the available resources, a more realistic decision could be taken based on this model.

The static properties used for a placement decision must be amended with a continuous monitoring of the actual performance data as the network load can be very dynamic. This can then form the basis for a potential VM migration decision. Within the CACTOS project [3], the monitored performance data is fed into an analytics framework and is used to build the basis for an optimisation framework. This optimisation framework not only supports initial placement decisions but also pro-active migration decisions and thus balances between performance benefits and migration costs. The work presented here, is an amendment to the existing system model of CACTOS. It predicts the performance of VMs on different types of compute hosts with a more accurate cut-off bandwidth compared to theoretical peak bandwidth of the installed network interface cards.

V. Conclusion and Outlook

In this paper, we have presented initial results to develop a model that is able to provide decision support for VM placement algorithms in a very short time for selecting an appropriate server to deploy a VM. Furthermore, the models can be used to make a buying decision for a given hardware. In order to determine a good balance between CPU/Memory resources and network capabilities, and to avoid imbalanced server design for Cloud data centres with communication intensive VMs, we can apply the communication behaviour of the models and the virtual switch models.

Further work is needed, in particular to validate if the modelling of VM traffic based on change state probabilities and corresponding rates fits to common workloads. Furthermore, we need to check if the virtual switch model, focused on the saturation bandwidth, is complex enough to properly emulate the behaviour of the different solutions for a wide range of hardware settings. This applies in particular for the tests with Open vSwitch, as the lack of performance of the CPUs in the MicroServer testbed in combination with 10GbE NICs, had significant impact on the overall performance. Also, we observed large variations in the performance for different minor software revisions and kernel versions. Additional comparisons between different virtual switch implementations using different architectural approaches and their effects on available host network resources, are planned for evaluation as well.

Acknowledgement

This research was supported in part by the bwCloud and bwNET100G+ projects funded by the Ministry of Science, Research and the Arts Baden-Wrttemberg (MWK) and the European Unions Seventh Framework Programme funded project CACTOS under grant agreement 610711.

References

[1] C. Loken, D. Gruner, L. Groer, R. Peltier, N. Bunn, M. Craig, T. Henriques, J. Dempsey, C.-H. Yu, J. Chen et al., "Scinet: lessons learned from building a power-efficient top-20 system and data centre," in Journal of Physics: Conference Series, vol. 256, no. 1. IOP Publishing, 2010, p. 012026.

[2] HPC System "JUSTUS" for computational chemistry. (visited on 2016.01.11). [Online]. Available: http://www.bwhpc-c5.de/wiki/index.php/Hardware_and_Architecture_(bwForCluster_Chemistry)

[3] S. Wesner, H. Groenda, J. Byrne, S. Svorobej, C. Hauser, and J. Domaschka, "Optimised cloud data centre operation supported by simulation," in eChallenges e-2014, 2014 Conference, Oct 2014, pp. 1–9.

[4] "Big data meets high performance computing," Intel corporation, Tech. Rep., 2014.

[5] S. Graf and F. Jülich, "Entwicklung eines werkzeugs zur analyse des os-jitter effekts auf high-performance cluster-systemen," Master's thesis, 2009.

[6] J. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering," in INFOCOM, 2012 Proceedings IEEE, March 2012, pp. 2876–2880.

[7] R. Cohen, L. Lewin-Eytan, J. Naor, and D. Raz, "Almost optimal virtual machine placement for traffic intense data centers," in INFOCOM, 2013 Proceedings IEEE, April 2013, pp. 355–359.

[8] A. Gupta, D. Milojicic, and L. V. Kalé, "Optimizing vm placement for hpc in the cloud," in Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit. ACM, 2012, pp. 1–6.

[9] Open vSwitch. (visited on 2016.01.11). [Online]. Available: http://openvswitch.org/

[10] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar et al., "The design and implementation of Open vSwitch," in 12th USENIX Symposium on Networked Systems Design and Implementation, 2015.

[11] M. Sarker, "Network infrastructure concept supporting fault tolerance cloud service provision," Master's thesis, Ulm University, Institute of Information Resource Management, 2015.

[12] DPDK. (visited on 2016.01.11). [Online]. Available: http://dpdk.org/

[13] 6WIND, "6WIND Support for Intel Data Plane Development Kit (DPDK)," 2014.

[14] Intel Open Network Platform Server Reference Architecture (Version 1.1), Intel, 2014.

[15] Network Function Virtualization: Intel Data Plane Development Kit vSwitch with Linux Virtualization and Intel Architecture, Intel, 2014.

[16] Lagopus switch. (visited on 2016.01.11). [Online]. Available: https://lagopus.github.io/

[17] Kazuaki Obana, NTT Network Innovation Laboratories, "SDN software switch Lagopus and NFV enabled software node," 2014.

[18] Mellanox OpenStack Solution Reference Architecture, Mellanox Technologies, 2013.

[19] Mellanox CloudX, Mirantis Fuel 5.1 / 5.1.1 Solution Guide, Mellanox Technologies, 2014.

[20] OpenStack Community. OpenStack Open Source Cloud Computing Software. (visited on 2016.01.11). [Online]. Available: https://www.openstack.org/

[21] Jon Dugan, Seth Elliott, Bruce A. Mah, Jeff Poskanzer, Kaustubh Prabhu. iPerf - The TCP, UDP and SCTP network bandwidth measurement tool. (visited on 2016.01.11).

 ISBN: 978-1-61208-450-3

APPENDIX A
OPENSTACK ENVIRONMENT SPECIFICATION

OpenStack Hosts	
CPU	2x Intel Xeon E5-2630 v3 @ 2.40Ghz
RAM	16x 16GB DDR4-2133 DIMM REG ECC 2R
Storage	2x SSDs in RAID-1 for OS
Storage	HDD based Ceph with separate SSD cache for VMs
OS	CentOS 7 (current patch level as of the time of the measurements)
NIC	Mellanox ConnectX3-Pro 1-Port 56GbE QSFP, MTU set to 1500
Switch	Mellanox SX1012 12-port 56GbE QSFP switch
Cable	Mellanox QSFP+ 1m passive copper 56GbE capable cable
Kernel	3.10.0-229.11.1.el7.x86_64
vSwitch	Open vSwitch version 2.3.1

OpenStack Instances	
CPU	2 vCores
RAM	4GB
Storage	10GB disk (Nova RBD backend)
OS	CentOS 7 GenericCloud
NIC	eth0: flat external network, MTU 1450
NIC	eth1: VXLAN tenant network, MTU 1450

APPENDIX B
ENCAPSULATION ENVIRONMENT SPECIFICATION

Test Bed 1		
Hardware and software	Number	Specification
Personal computer	2	CPU Intel(R) Core(TM) i54670 CPU @ 3.40GHz, Memory 16GiB, OS Ubuntu 14.04, Kernel version 3.14.27-031427-generic (PC1), 3.13.0-44-generic (PC2)
Switch with 1GbE NIC	1	NETGEAR, Series ProSafe, Model GS108v3, Network interface RJ-45 connector for 10BASE-T, 100BASE-T, or 1000BASE-T Ethernet interface
Ethernet cable	1	Logilink U/UTP CAT6 24AWGX4P PATCH ISO/IEC 11801 and EN 50173 VERIFIED
Open vSwitch	1	version 2.0.2

Test Bed 2		
Hardware and software	Number	Specification
MicroServer	2	CPU Intel(R) Celeron(R) CPU G1610T @ 2.30GHz, Memory 2GiB, OS CentOS 7, Kernel version 3.10.0-123.20.1.el7.x86_64
10GbE NIC	2	HP Ethernet 10Gb 2-port 530SFP+ Adapter, Network Processor QLogic 57810S chipset, Data Rate Two ports, each at 20 Gbps full duplex; 40 Gbps aggregate full duplex theoretical bandwidth.
10GbE cable	2	HP X242 SFP+ SFP+ 3 m Direct Attach Cable (J9283B), Length 10 ft. (3 m)
Open vSwitch	1	version 2.3.1

 ISBN: 978-1-61208-450-3