An Experimental Evaluation of Lightweight Virtualization for Software-Defined Routing Platform

Carlos N. A. Corrêa, Sidney C. de Lucena
and Daniel de A. Leão Marques
Federal University of the State
of Rio de Janeiro (UNIRIO)
Rio de Janeiro - RJ - Brazil
{carlos.correa,sidney,daniel.marques}@uniriotec.br

Christian E. Rothenberg
and Marcos R. Salvador
Telecomm. Research and Development
Center (CPqD)
Campinas - SP - Brazil
{esteve,marcosrs}@cpqd.com.br

***Abstract*—Network virtualization is one of the foundations for Future Internet architectures, able to catalyze technologies for the solution of classical problems in the IP layer. The work here gives a step forward in that direction, evaluating the performance of different virtual network tools under the perspective of a virtualized IP routing control plane. Two lightweight container-based virtualization systems, OpenVZ and LXC, were selected based on specific criteria and evaluated in the context of virtual routers. Obtained results allow for a comparative analysis about the convergence of the routing protocol and give insights about the adoption of these tools on a routing as a service platform.**

I. Introduction

Cloud computing and virtualization in general provide a flexible and rational utilization of networked computer resources. Even more, the fundamental underpinning is the provision of a software management layer that allows for a programmatic definition of the entire infrastructure tailored to the needs of the tenants' applications. As the virtualization paradigm permeates from computer to networking technology, the opportunity to let software flexibility change the ancient and rigid networking functions is set.

OpenFlow [1] is one enabling technology of the so-called Software Defined Networking (SDN) approaches, that advocates for new networking abstractions by means that include a clean separation of the operation of data forwarding equipment (the data plane) from it's control logic (the control plane), which can be hosted in a new network controller element.

The physical separation and logical centralization of the control functions allow the migration of such components to service-oriented scenarios. As such, SDN-based virtual IP routing platforms like DROP [2] and RouteFlow [3] can be used to materialize concepts like Plataform-as-a-Service (PaaS) for networking and single router abstraction [4]. This can result in new perspectives to routing operation, or even it's separation from routers [5].

The capabilities of the virtualization plane components are paramount to the performance and functional flexibility of these split architectures. So far, previous work on the verge of virtualized versions of traditional routing has not brought the control/forwarding plane separation into consideration [6]. This work presents a performance evaluation of different network virtualization tools under the perspective of a virtualized IP routing control plane. Nevertheless, it aims to provide guidance for the advance of other solutions of the same kind that seek to leverage server virtualization technologies to develop instantiations of the SDN paradigm.

The balance of the paper is as follows. Section 2 details the general framework for this type of systems operations. The functional requirements for such components are analyzed in Section 3 Section 4 describes the adopted methodology for the evaluation of candidate virtualization tools and Section 5 discusses the results of that experimental work. Finally, Section 6 presents the conclusions and work ahead.

II. Virtualization and forwarding planes separation

There are mainly two ways in which virtualization can be done: the full virtualization, where each guest runs its own operating system (OS), and the lightweight virtualization based on containers or processes, where the host's OS shares and isolates the available resources among the guests' systems [7].

As much as system virtualization is about computer resources sharing, network virtualization (NV) permits the instantiation of multiple logical networks over the same physical infrastructure. In [8], four approaches are presented for the implementation of NV: (i) Virtual Local Area Networks (VLANs), (ii) Virtual Private Networks (VPNs), (iii) active/programmable networks, and (iv) overlay networks.

The definition of a standard API to networking hardware such as OpenFlow [1] lays ground for a fifth approach. OpenFlow-enabled switches use this standard protocol to interact with network controller elements. Typically, the controller includes software logic to analyze packet headers of data flows and instructs switches on how to handle these flows – either in a proactively or reactively fashion.

978-1-4673-0269-2/12/$31.00

Virtual IP routing platforms derive from the feasibility of incorporating additional logic to devices previously dedicated to switching. In these architectures, routing, originally handled by routers in a distributed fashion, are now handed over by switches to a controller platform. The controller queries a virtualized (and eventually abstracted) version of the network, for which virtualizers and virtual connectivity tools are both employed. Routing information is derived from the virtual network, and in turn transformed into instruction sets that determine how traffic packets should be handled by switches.

While the functions performed by virtualization tools in the context of virtual routing platforms are well known, we have not found any thorough analysis to justify the choice for tools to compose a virtualized control plane, both from functional and performance points of view. This condition makes comparative studies difficult and limits their use outside experimental environments.

III. Requisites of an IP routing services platform

After studying several IP routing platforms and the roles played by virtualizers in each one of them, this work establishes the following requirements for this kind of component:

- **Isolation.** The virtualization software should be able to segregate the basic resources and the network stack of the systems running under its supervision.
- **Efficiency.** The tool should be able to cause the lowest possible overhead in order to keep the maximum of the host's resources available to the VMs.
- **Scalability.** The relationship between the consumption of system resources and the number of running VMs should be approximately linear, giving enough elasticity for the number of nodes that can be created in the virtual plane.
- **Flexibility.** The virtualization tool must support the attachment of multiple virtual network adapters for each VM.

Requirements were also identified for the virtual connectivity components. Some of them are similar to the ones listed for virtualization software such as: **Isolation**, **Efficiency**, **Scalability** and **Flexibility**. The others identified are:

- **Multiple access.** It should be possible to connect multiple virtual network adapters by means of a single virtual connectivity device.
- **Extensibility.** Preferentially, it should be possible to extend the connectivity provided by mechanism in order to support a distributed control plane.

After defining requirements for the components of virtual IP routing platforms, we need to select those with the best fit.

We identified, from the literature, the Xen hypervisor [6] and the tools for lightweight virtualization based on containers: Linux-VServer [7] and OpenVZ [9]. Also there is an aditional candidate, LXC [10], who has not been applied at virtual IP routing platforms yet.

Xen is a proven technology (cf. [6]) for the deployment of virtualized version of traditional routers. However, considering the architectures where the control plane operation is entirely virtualized, a full-featured hypervisor concurs against the efficiency requirement.

Linux-VServer does not fully support host's IP stack virtualization, which affects the isolation requirement and limits its use in the solution space envisioned.

LXC is a lightweight virtualization module recently integrated into the Linux kernel that includes network-stack virtualization feature, which qualifies it for the isolation aspect.

Consequently, from a functional point of view, the candidate tools that match the proposed requirements are the lightweight virtualization technologies OpenVZ and LXC.

For the role of virtual connectivity, beyond the TUN/TAP framework, widely used by the proposals in the literature, candidate technologies include the Linux *bridges* and the software-based switch Open vSwitch, or simply OVS.

The TUN/TAP framework, however, applies only to the creation of virtual interfaces for point-to-point communications, which contradicts the requirement for multiple access and disqualifies it for the purposes of this evaluation. Both the Linux bridges and OVS comply to the functional aspects of isolation, flexibility and extensibility established for this kind of component.

IV. Experimental evaluation

For the evaluation, each node of the control plane corresponded to a virtual router running the Quagga routing suite with only the OSPF daemon activated. In order to provide data relevant to the most demanding operation mode, OSPF message interval (Hello Interval) was set to 1 second. The unavailable link detection interval was consistently adjusted to 4 seconds.

Two types of topologies were explored, corresponding to different scalability requirements:

- **Grid-based topologies:** Two network topologies were defined, with nodes arranged in a 3x3 and a 4x4 grid.
- **Full-mesh topologies:** Three fully-meshed network topologies of 15, 25 and 35 nodes were evaluated to focus on the actual behaviour of the OSPF protocol.

The connectivity between the virtual routers provided by Linux bridges and OVS was also tested. The combination of OpenVZ hypervisor with OVS connectivity was not evaluated due to tool incompatibility at that time.

The low-scale set of experiments was run on a Intel Core 2 Duo 2.93Ghz platform with 3GB of RAM, and the second set of experiments was executed on a system with an Intel Xeon X5660 CPU and 48GB of RAM, both running Debian GNU/Linux 5.0 and dedicated to the experiment tasks.

Three different stages in the operation of the virtual control plane were introduced for evaluation: (i) initialization, (ii) regular operation (ten minutes after the initialization of each virtual topology), and (iii) topology changes (i.e., link down and link up events).

After system's initialization overhead, the measurement of the regular operation stage helps quantifying the resource consumption of tools in the absence of connectivity change events. The topology changes phase allows to characterize the

TABLE I
AVERAGE BOOT-TIME OF NODES ON GRID TOPOLOGIES

Virtualization	Connectivity	Topology	Initialization Time (s)
LXC	Bridge	3x3	31
		4x4	62
	OpenVSwitch	3x3	32
		4x4	63
OpenVZ	Bridge	3x3	39
		4x4	74

TABLE II
AVERAGE MEMORY USAGE ON GRID TOPOLOGIES

Tools	Topol.	RAM Utilization (MB)		
		Initialization	Regular Op.	Disconn.
LXC+Bridges	3x3	169	180	206
	4x4	259,50	276	314
LXC+OVS	3x3	175	185	228
	4x4	269,75	286	369
OpenVZ+Bridges	3x3	180	226	268
	4x4	337,25	350,5	693

efficiency of the components tested relative to the occurrence of events in the data plane, resulting in the exchange and processing of LSAs. For stress test purposes, simultaneous failures of links in the grid topologies were introduced until reducing the networks' connectivity graph to a minimum spanning tree of itself. On full-meshed cases, a failure of all links to a single node was introduced. In both cases, the same links were re-established 120 seconds later after disconnection, so that different categories of events were observed in experiments.

The following data of interest were collected:

- **Boot-time of the nodes.** Time interval needed for nodes to be fully operational.
- **CPU and memory usage.** Allow to infer the degree of overhead imposed by each virtualization solution.
- **Convergence time.** Measures the convergence time of the OSPF protocol at different stages. During initialization, the measured time is the interval between topology boot up and the first convergence of the OSPF protocol.

In order to confer statistical relevance to the obtained results, each metric had its data collected through 30 distinct executions of each network topology for each scenario.

V. ANALYSIS OF THE RESULTS

Table I presents the results for the initialization time of the topologies. LXC with bridges-based connectivity is the fastest, performing the task with an average time 20% lower than OpenVZ for 3x3 grids. For 4x4 grids, the difference is 16%. Both relations keep respectively at 18 and 15% when OpenVZ is compared to LXC's VMs connected via OVS.

Figure 1a shows the average CPU usage during VMs' initialization on grid topologies. It shows that both LXC and OpenVZ lead to 100% CPU usage during the initialization process. However, LXC faster initialization puts it on advantage, since processor utilization decays more quickly. LXC results using both OVS and bridges fall in the confidence range of each other, so the former are not represented.

Table II presents the average memory use of evaluated tools, for all the grid scenarios. Through the average use of RAM during VM's initialization, it is possible to find that both LXC and OpenVZ show an evolution of consumption with the increasing scale of topology - with significant advantage for the LXC demand. OVS represents an additional demand of 2,8 to 4% of RAM over bridges.

As shown in Figure 2a, the measured time for the initial convergence on grid topologies presents results very different from what would be expected by looking just to Table I. This time OpenVZ has the best performance, with LXC showing a delay between the instantiation of virtual networks (independently of connection type) and the beginning of the OSPF data exchange. During the tests, even in the fastest-loading LXC grid topologies, OSPF announces were exchanged only 50s after the start of the experiment. In contrast, OpenVZ starts the OSPF exchange almost immediate after experiment initialization.

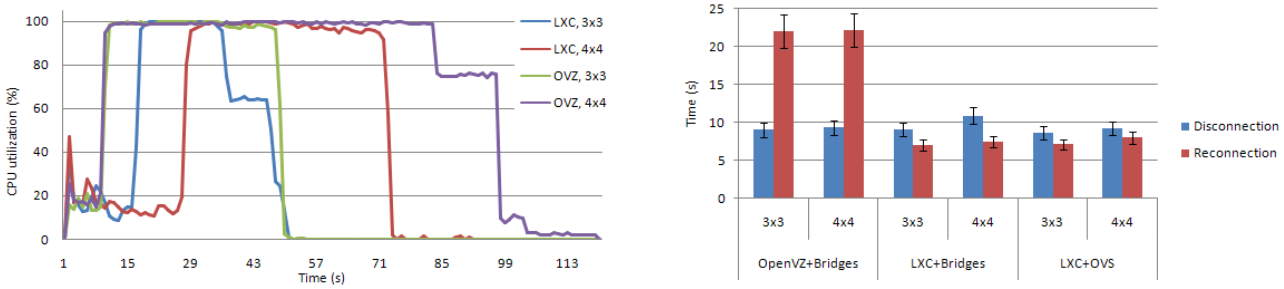
However, this delay could be explained by differences in the way OVZ and LXC network interfaces are started. For the software versions evaluated, on OVZ a VM network device can be defined as being part of a bridge, while on LXC one needs to wait for the initialization of a container before adding one of it's interfaces to a bridge or OVS instance. As such, data depicted in Figure 1b shows that convergence after connectivity modification events is not affected by the delay in the start of the operation of the virtual networks. The disconnection times are very close, but, for reconnection times, the OpenVZ performs worse than the LXC combinations.

The CPU usage does not achieve more than 3% during the realization of the tests with link events. According to the results in Table II, the memory usage grew in this period (disconnection). In the OpenVZ 4x4 topology, the usage of RAM is emphasized in the occurrence of failures, which might indicate a scalability issue. In the absence of new events, the observed RAM usage returned to average in all cases.

Average times for initial convergence on full-meshed networks are presented on Figure 2b, showing again best results for OpenVZ. It is also faster on after-disconnection convergence tests (shown in Figure 3a), being surpassed by the LXC+OVS combination only in the case of convergence time after reconnection (Figure 3b).

VI. CONCLUSION

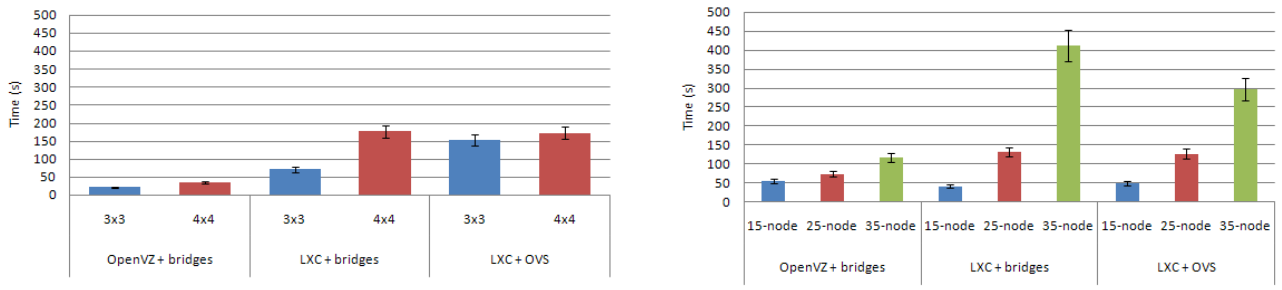
The results show LXC topologies' initialization is faster than on OpenVZ. They also suggest that the specifics of container definition in the software versions tested put in advantage OpenVZ's initial OSPF convergence. Nonetheless, for OSPF convergence after connectivity modification events, OpenVZ scores more than double the time necessary for convergence under LXC in grid scenarios. The full-mesh topologies show mixed results, but LXC+bridges combination is clearly the least performant. Finally, LXC+OVS combination has a smaller memory footprint than OpenVZ, while offering the flexibility of an OpenFlow-enabled and potentially distributed control plane.



(a) CPU utilization

(b) Convergence after connectivity modification events

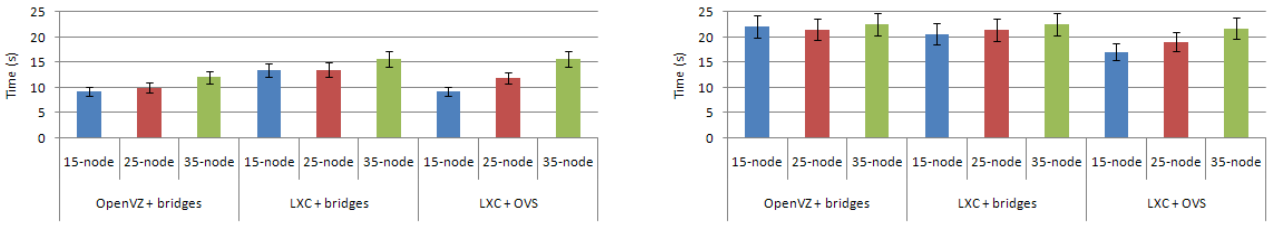
Fig. 1. Grid topologies' CPU use and link events convergence details



(a) Initial convergence on grid topologies

(b) Initial convergence on full-meshed topologies

Fig. 2. Initial OSPF convergence for both connectivity layouts evaluated



(a) Convergence after disconnection of a node

(b) Convergence after reconnection of node

Fig. 3. Convergence on full-meshed networks after connectivity modification events

Overall, we conclude LXC+OVS is the best option for a virtual control plane topology, but optimizations for LXC's startup are being planned for future works.

References

[1] 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.*, vol. 38, pp. 69–74, March 2008.

[2] R. Bolla, R. Bruschi, G. Lamanna, and A. Ranieri, "Drop: An open-source project towards distributed sw router architectures," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, 302009-dec.4 2009, pp. 1 –6.

[3] M. R. Nascimento, C. E. Rothenberg, M. R. Salvador, C. N. A. Corrêa, S. C. de Lucena, and M. F. Magalhães, "Virtual routers as a service: the routeflow approach leveraging software-defined networks," in *Proceedings of the 6th International Conference on Future Internet Technologies*, ser. CFI '11. New York, NY, USA: ACM, 2011, pp. 34–37.

[4] E. Keller and J. Rexford, "The "platform as a service" model for networking," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, ser. INM/WREN'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 4–4.

[5] M. Caesar, M. Casado, T. Koponen, J. Rexford, and S. Shenker, "Dynamic route recomputation considered harmful," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 66–71, April 2010.

[6] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, L. Mathy, and P. Papadimitriou, "A platform for high performance and flexible virtual routers on commodity hardware," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 127–128, January 2010.

[7] S. Bhatia, M. Motiwala, W. Muhlbauer, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford, "Hosting virtual networks on commodity hardware," Tech. Rep., 2008.

[8] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, pp. 862–876, April 2010.

[9] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: live router migration as a network-management primitive," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 231–242, August 2008.

[10] LXC, "lxc linux containers - container namespace cgroup virtualisation," 2011, `http://lxc.sourceforge.net/`.