Scalability of Container Overlays for Policy Enforcement in Digital Marketplaces

Shakeri, S.; van Noort, N.; Grosso, P.

DOI
10.1109/CloudNet47604.2019.9064090
Publication date
2019
Document Version
Final published version
Published in
2019 IEEE 8th International Conference on Cloud Networking (CloudNet 2019)

Scalability of Container Overlays for Policy Enforcement in Digital Marketplaces

Sara Shakeri
Systems and Networking Lab
University of Amsterdam
Amsterdam, The Netherlands
s.shakeri@uva.nl

Niek van Noort
University of Amsterdam
Amsterdam, The Netherlands
niek.vannoort@student.uva.nl

Paola Grosso
Systems and Networking Lab
University of Amsterdam
Amsterdam, The Netherlands
p.grosso@uva.nl

***Abstract*—Digital marketplaces (DMPs) are emerging as a framework for organizations to share their data. Security and support for multi-tenancy are the key features of DMPs. DMPs infrastructure can be built upon container-based networks in the cloud environments. However, there is not at the moment an in-depth analysis of the capability of container networks to support this mode of operation. In this paper, we evaluate the capability of Cilium and Calico, the two most popular container network techniques, in providing security (*policy scalability*) and handling the multi-tenancy requirements (*pod scalability*) of DMPs. We first measured the policy scalability in the network, and both Calico and Cilium scale well. However, by studying the pod scalability we determine there is around 50% throughput degradation in both technologies by increasing the number of pods from one to forty.**

***Keywords*—Digital Marketplace, Data Sharing, Containers, Docker, Container Overlay Networks, Cloud Environment.**

I. Introduction

Digital marketplaces [1] (DMPs) constitute a novel framework for secure data sharing and they are governed by rules and agreements among participating parties. In fact, different organizations are willing to share their data with each other only if the data exchanges follow predefined rules. DMPs guarantee exactly this: that the digital collaboration between multiple tenants is efficient and secure. Therefore, providing security and handling multi-tenancy are of the most important requirements which should be provided in a secure DMP.

The participating organizations in a DMP want to use both shared information and shared computational applications [2]. Therefore, two kinds of digital resources can be shared among organizations in a DMP: algorithm and data. Fig. 1 depicts the general architecture of a secure DMP. All of the digital collaborations in a DMP are based on the access rights and agreements among participating organizations, i.e. algorithm suppliers and data suppliers [3]. The agreements can be presented in different description models [4]. They will be translated into the network policies which filter the allowed traffic flow. Enforcing these network policies in the network plays an important role in providing security in a DMP. Agreements between DMP parties need to be converted in deployment models and deployment specifications. Concretely, cloud systems can be used for setting up the DMP as they can provide the sharing platform for different organizations with much less time and cost. A container-based solution, e.g. Docker [6], can be deployed in the cloud to construct the sharing application platform. Docker containers can act as participating parties in a DMP and perform data sharing. To put the containers in connection with each other and enforce the network policies, one of the container overlay network technologies can be deployed [7]. There are different implementation methods of container overlay technologies and selecting the proper one is highly dependent on the application's workload and its requirements.

978-1-7281-4832-8/19/$31.00 ©

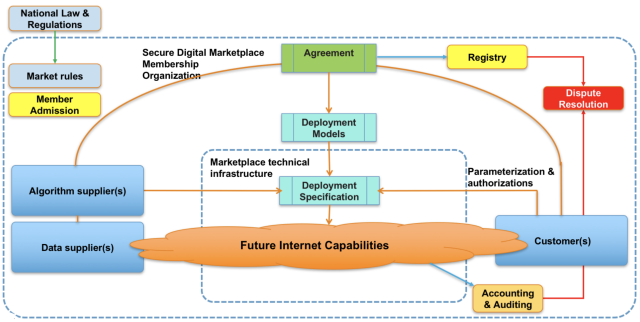

Fig. 1. Secure Digital Marketplace Framework [5]

In this paper, we specifically focus on the sharing application in a DMP and consider enforcing the network policies and supporting the multi-tenancy as its requirements. Our goal is evaluating how container overlay network technologies can fulfill these requirements. A lot of works have been done for analyzing different characteristics of container overlay networks [7]. However, to the best of our knowledge, there is no study which investigates the capability of container overlay technologies in sharing applications. To this end, we set up a container-based sharing platform for emulating a DMP and deployed Cilium [8] and Calico [9] as overlay network technologies, as they have the best support for enforcing network policies [10]. The network is set up as a Kubernetes cluster and containers are running inside the pods [11]. To evaluate Cilium and Calico performance in policy enforcement and handling the multi-tenancy, we observe how well these two overlay technologies scale with an increasing number of

policies and pods in the network.

II. Policy and Pod Scalability in DMPs

According to the agreements established in a DMP, various scenarios can be defined for sharing the data and algorithm. Four of them are depicted in Fig. 2. Each scenario determines the permitted and prohibited traffic flow among the organizations which can be translated to network policy and then implemented in the container-based infrastructure.

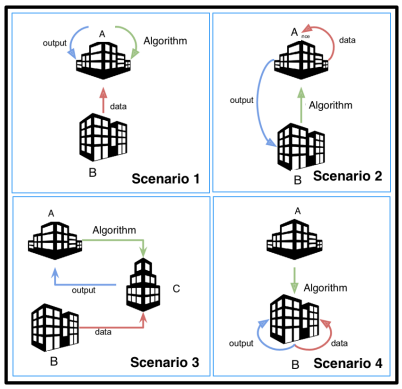

Fig. 2. Examples of sharing scenarios in a DMP

Let's consider Scenario 1 in Fig. 2 as the target scenario which should be implemented in the infrastructure. It shows the algorithm of organization A will be executed on data of organization B at the location belonging to organization A, and then organization A will use the output of the operation. To implement this, it is necessary to define a policy in the network which permits all of the connection from organization B to organization A and prohibits any other connection from other parties. Listing. 1 shows the Kubernetes network policy that accomplishes the desired behavior.

Listing 1
A container network policy, specifying connection rules between organization A and organization B

```
kind: NetworkPolicy
metadata:
  name: Network_Policy_Example
spec:
  podSelector:
    matchLabels:
      id.pod : OrganizationA
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          id.pod : OrganizationB
```

However, running the policy imposes an overhead in the network and may affect the network performance, especially when the number of policies increases. It becomes therefore very important to quantify the **policy scalability** (the throughput of the network when the number of policies increases) of each container networking technology.

A second other important factor is the network performance when the number of pods increases. In many cases in a Kubernetes multi-tenant cluster, it is necessary to run multiple numbers of pods at the same time to handle the applications' requests. Therefore, it is worth it to investigate the performance in handling concurrent communications between multiple pods, e.g. **pod scalability**.

III. Container Networking: Overlay Technology

The method of bringing the connectivity among containers across multiple machines can be provided by overlay networks. The overlay network can handle the access rules of the traffic flows by enforcing the policies. There exist various implementations of overlay networks for Docker containers which are also integrated with Kubernetes, like Weave [12], Flannel [13], Cilium, and Calico. The network policies can be defined and implemented by container overlay technologies. In this work, we investigate two of the most popular overlay networks technologies as Cilium and Calico.

Cilium – Cilium is an open source technology that is developed for securing the network connectivity of the Linux containers which are managing by Docker and Kubernetes. It leverages eBPF [14] as a technology for filtering and security policy enforcement. In the Cilium architecture, the Cilium agent will set up the connectivity and networking among containers in a cluster and also is responsible for deploying the network security policies. Linux kernel eBPF runs the bytecodes which are compiled by Cilium in order to enforce the security and policies over the traffic among containers from within the kernel. In Cilium, all the packets which are sent by a container to an endpoint in the overlay network, are encapsulated by VXLAN.

Calico – Calico is used to create overlay networks and establish connections between containers across the nodes. The Calico node agent consists of three main components: *felix*, *bird*, and *confd*. *felix* is responsible for providing the connectivity and policy enforcement by programming the routes and iptable on the host. *bird* distributes the routing information which is programmed by *felix* between hosts as a Linux BGP agent. In case of any changes to the BGP configuration in the etcd datastore *confd* triggers *bird* to reload the changes on each host. There are two methods for bringing the connectivity between multiple hosts in Calico: BGP and IPIP. In IPIP the original packet with the container IP addresses will be encapsulated at the network layer with the host IP address. In Calico, the policy will be translated to the host iptable rules.

IV. Experiment Setup

We have set up the sharing platform as a Kubernetes cluster, utilizing three VMs running on separate physical machines. The VMs are connected via 10Gbps Ethernet link and each VM is running Ubuntu 18.04 and Linux kernel 4.15 as the operating system and has access to one CPU core. One of the VMs is functioning as a master node. The other two VMs are joined to the cluster as worker nodes and run Kubernetes pods. We use Cilium version 1.6 and Calico version 3.5 as overlay network in the experiments. Fig. 3 depicts the network setup.

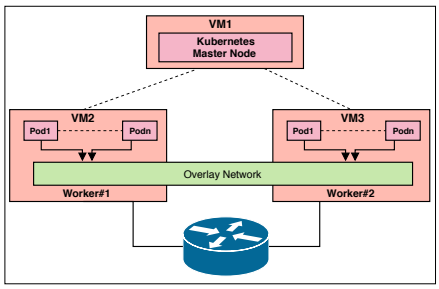

Fig. 3. Network experiment setup, emulating a DMP

V. Results

Policy scalability – We evaluate the network policy scalability by measuring the throughput of the network when the number of policies increases. The results are the mean throughput of three times experiment and each time for 300 seconds, measured by iperf3. Fig. 4 depicts the result of measuring the throughput of the network from a running pod in a worker node to another one, when the number of policies increases. The difference between the policies is the port number. The policies are set in the way that the incoming traffic will be matched with the last policy. As the results in Fig. 4 show, although there is little throughput degradation, overall both Calico and Cilium perform well in policy scalability.

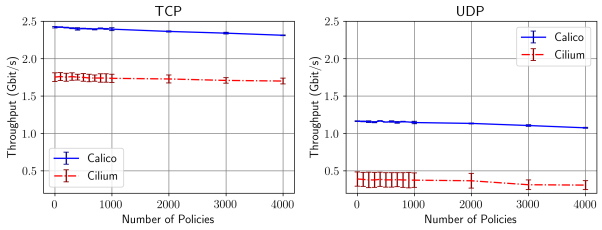

Fig. 4. The mean throughput as a function of the number of network policies

Pod scalability – The pod scalability will be tested by increasing the number of pods and measuring the throughput in the network. The results are the average of three runs and each time for 300 seconds using iperf3. In every single experiment carried out in our setup (see Fig. 3), we run a specific number of pods in one VM functioning as a client and in the other one as a server. Each time, we have calculated the total throughput that is achieved in the network and compared it to the expected throughput. Expected throughput is the maximum throughput that can be gained in the network running one client-server stream.

The result of pod scalability is depicted in Fig. 5. As it shows Calico throughput is more than Cilium. However, there is throughput degradation in TCP and UDP traffic for both Cilium and Calico. For TCP, Calico has 1.2 Gbps throughput loss, and Cilium has 1 Gbps drop for 40 pods. Also, in UDP the throughput drop is 0.3 Gbps for Calico and 0.1 Gbps for Cilium. As a result, both Calico and Cilium do not scale well with increasing the number of pods. This behavior is caused by increasing the number of connections to the master node and making a bottleneck in the network. In addition, as the observations show, TCP scales worse than UDP in both Cilium and Calico. TCP has the Ack handling process which UDP does not have, and this causes more interrupts in the CPU and network interface so that we observe more throughput reduction in TCP.

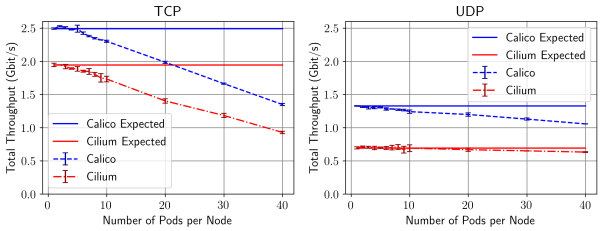

Fig. 5. The cumulative throughput as a function of the number of pods

VI. Comparison

There are two major implementation differences between Cilium and Calico:

Encapsulation method – To provide an overlay network Cilium uses VXLAN technology, Calico uses IPIP. VXLAN has more overhead for packets's generation compared to IPIP. On the other hand, IPIP is not supported in all infrastructures and cloud environments.

Policy translation method – Cilium uses BPF program running inside the host kernel for policy enforcement. It defines the identity for each pod based on its label and then uses a hash table for storing the policies. Calico will translate the policies to the iptable rules of the host.

As we saw, Calico has always better throughput than Cilium. This is mainly because of the different encapsulation method. Calico uses IPIP which imposes less overhead than VXLAN. In fact, VXLAN will increase the packet size more than IPIP and consequently, because of the limited MTU in the network interface, the packet will be fragmented. This increases the CPU usage in the host and creates additional interrupts in the network interface. Moreover, both Calico and Cilium lose some throughput when the number of policies increases (see Fig. 4). The throughput drop in Calico is higher than Cilium as we increase the number of policies. The reason is the difference in the policy translation method. In Cilium increasing the number of policies does not affect the filtering process due to the fact that Cilium checks the policies for every packet with a hashtable lookup of complexity O(1). However, in Calico as the rules are translated to the host iptable rules, more rules should be checked with increasing the number of the policies and this will negatively affect the throughput. Therefore, the throughput reduction will be more in Calico than that in Cilium.

VII. Discussion

Investigating more on the reasons for the lower performance of Cilium, we have offloaded the etcd pods running in Cilium

to another VM and repeated the experiments. Fig. 6 shows the experiment results. It shows Cilium's throughput is closer to the throughput of Calico by etcd offloading, but keep in mind that Cilium needs an extra node to accomplish etcd offloading.

In addition, Cilium doesn't reassemble fragmented UDP datagrams before making a decision about accepting or dropping them. As the first packet is the only one which has the UDP header information, the fragmented packets will be dropped by Cilium. To compare Cilium and Calico with equal datagram size, we measure Calico's throughput with a 1422 bytes datagram size. As the results in Fig. 6 show we verify throughput degradation in Calico with 1422 bytes datagram size, however, its performance is still better than Cilium.

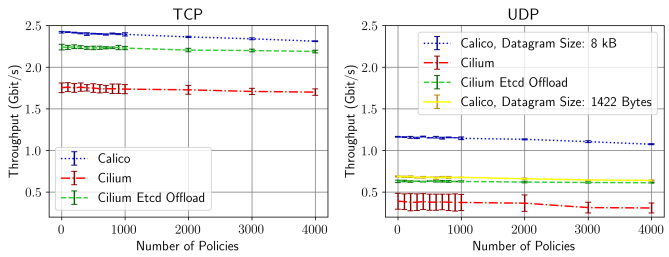

Fig. 6. The mean throughput as a function of the number of network policies deploying etcd offloading and UDP datagram size

VIII. Related Work

There are multiple works that investigate container network technologies and evaluate their performance. For example, in [7] authors presented an empirical study about different methods of container networks. They conducted a qualitative comparison of available methods regarding their different levels of isolation and overhead. Then, they have done lots of experiments for evaluating the performance of different container networks. [15] evaluated three technologies that support addressing and filtering in container overlays: EVPN, ILA, and Cilium. In addition, the authors presented the performance analysis of Cilium/eBPF in network filtering, specifically in multi-tenant environments.

Above studies have considered different implementations of container networking and evaluate each of them. However, to the best of our knowledge, there is no study which considers the capability of container overlay networks in sharing platforms.

IX. Conclusion

In this paper, we present an in-depth analysis of the capability of deploying the container networks specifically in DMPs for achieving secure and high-performance data sharing platform. We use the metrics policy scalability and pod scalability to evaluate if Cilium and Calico are suitable for deployment in a sharing platform. Calico and Cilium are built upon different architectures and their method in encapsulation, deploying the network policies and packet filtering is different. Both Cilium and Calico scale well in policy scalability as a function of the number of policies in the network. In fact, the number of policies applied to a cluster has little effect on the throughput, especially when Cilium is used. However, there is a substantial throughput degradation in both technologies in pod scalability. This introduces a challenge for deploying these technologies when there is a requirement for running a large number of pods. Thus, it depends on the application of the data sharing platform whether the throughput losses are acceptable. Overall, Calico has better throughput in all experiments. Therefore, on the scale experimented within this research, Calico would be the better choice for a secure data sharing platform.

As future work, we intend to continue the evaluation of other important container overlay network technologies like Weave and Flannel. We will also investigate the possible vulnerabilities of container-based networks in providing the security in sharing platforms. Then, we set up a container-based sharing environment deploying the best matched overlay network technology.

X. Acknowledgment

This work is supported by the Netherlands eScience Center and NWO under the project SecConNet.

References

[1] S. van den Braak, S. Choenni, R. Meijer, and A. Zuiderwijk, "Trusted third parties for secure and privacy-preserving data integration and sharing in the public sector," in *Proceedings of the 13th Annual International Conference on Digital Government Research*. New York, NY, USA: ACM, 2012, pp. 135–144.

[2] D. Thilakanathan, S. Chen, S. Nepal, R. Calvo, and L. Alem, "A platform for secure monitoring and sharing of generic health data in the cloud," *Future Generation Computer Systems*, vol. 35, pp. 102 – 113, 2014, special Section: Integration of Cloud Computing and Body Sensor Networks; Guest Editors: Giancarlo Fortino and Mukaddim Pathan.

[3] D. Harris, L. Khan, R. Paul, and B. Thuraisingham, "Standards for secure data sharing across organizations," *Comput. Stand. Interfaces*, vol. 29, no. 1, pp. 86–96, Jan. 2007.

[4] S. Shakeri, V. Maccatrozzo, L. Veen, C. de Laat, and P. Grosso, "Modeling and matching digital marketplace policies," accepted to eScience 2019 workshop.

[5] L. Zhang, R. Cushing, L. Gommans, C. De Laat, and P. Grosso, "Modeling of collaboration archetypes in digital market places," *IEEE Access*, vol. 7, pp. 102 689–102 700, 2019.

[6] "Docker," https://www.docker.com/, 2019, [Online; accessed June-2019].

[7] K. Suo, Y. Zhao, W. Chen, and J. Rao, "An analysis and empirical study of container networks," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, April 2018, pp. 189–197.

[8] "Cilium," https://docs.cilium.io/en/v1.5/, 2019, [Online; accessed June-2019].

[9] "Calico," https://docs.projectcalico.org/v2.0/introduction/, 2019, [Online; accessed June-2019].

[10] "Tigera," https://www.tigera.io/media/pr-best-product-container-security/, 2019, [Online; accessed June-2019].

[11] "Kubernetes," https://kubernetes.io/docs/tutorials/kubernetes-basics/, 2019, [Online; accessed June-2019].

[12] "Weave Net," https://www.weave.works/oss/net/, 2019, [Online; accessed May-2019].

[13] "Flannel," https://github.com/coreos/flannel#flannel, 2019, [Online; accessed June-2019].

[14] "The Berkely Packet Filter," https://www.kernel.org/doc/html/latest/bpf/index.html, 2019, [Online; accessed June-2019].

[15] L. Makowski and P. Grosso, "Evaluation of virtualization and traffic filtering methods for container networks," *Future Generation Computer Systems*, vol. 93, pp. 345 – 357, 2019.