# Análise Empírica do Desempenho de Redes de Comunicação em Contêineres: um Estudo Preliminar

**Lucas Litter Mentz**[1], **Guilherme Piêgas Koslovski**[1]

[1]Universidade do Estado de Santa Catarina (UDESC) – Joinville – SC – Brasil

lucaslittermentz@gmail.com, guilherme.koslovski@udesc.br

***Resumo.*** *O encapsulamento de aplicações de alto desempenho em contêineres permite melhores desempenhos em cenários limitados por memória ou processamento, entretanto limita a liberdade de conectividade entre as tarefas. Para assemelhar a comunicação de contêineres à de redes convencionais são usados network drivers específicos. Neste trabalho são revisados os principais drivers de rede padrão do Docker e uma coleção de trabalhos abordando seus desempenhos. Por fim, uma proposta de análise empírica de desempenho das redes de comunicação em contêineres Docker é apresentada.*

## 1. Introdução

Contêineres são uma forma de virtualização leve com base em particionar o sistema operacional hospedeiro em ambientes isolados logicamente. Entre as vantagens de contêineres sobre virtualização de *hardware* estão menor sobrecarga de processamento, memória e armazenamento devido a compartilhar o sistema operacional do hospedeiro ao invés de cada um executar uma instância completa [Vaughan-Nichols 2006]. Contêineres são usados para criar desde pequenos cenários portáteis de desenvolvimento até aplicações escaláveis, de alto desempenho e resistentes a falhas.

Contêineres dedicam seus pontos positivos à sua implementação, isolamento em processos Linux. Em contrapartida às vantagens de desempenho, isso põe um obstáculo na comunicação entre contêineres. Enquanto a virtualização tradicional fornece às máquinas virtuais uma interface de rede virtual que para o sistema hóspede aparenta uma interface de rede real, contêineres necessitam de soluções engenhosas devido aos processos serem limitados a soquetes na rede. Para aproximar o funcionamento das redes de contêineres ao das redes comuns são utilizadas soluções denominadas *network drivers* que abstraem redes de contêineres para prover diferentes formas de conectividade. Como propõem formas de comunicação que não são nativamente suportadas por processos, os *drivers* requerem processamento extra.

Este trabalho discute *network drivers* Docker e apresenta um plano de testes para avaliar o impacto dessas soluções comparado ao uso de soquetes. Devido a larga utilização de contêineres e a degradação de desempenho de comunicação observada, este trabalho se propõe a identificar experimentalmente a carga adicional requerida pelas redes de contêineres para Docker e os impactos causados por estas. Espera-se que os resultados dos testes indiquem parâmetros aos quais os *drivers* estudados melhor se destacam.

## 2. Motivação e Definição do Problema

Existem diversas implementações populares de contêineres, entre estas Linux Containers (LXC), OpenVZ e Docker, sendo o último a implementação abordada neste trabalho.

Docker traz por padrão cinco *drivers* de rede: *none*, *host*, *bridge*, *macvlan* e *overlay* [Docker, Inc. 2019]. Para cada *driver* Docker é esperado um comportamento diferente, inerente aos processos envolvidos.

Docker *none* não implementa comunicação de rede. Docker *host* é o *driver* padrão e implementa somente o equivalente a soquetes. Docker *bridge* cria uma ponte e permite aos contêineres ligados à essa ponte usarem uma sub-rede com direito a endereçamento válido nela. Docker *macvlan* liga o contêiner à uma porta virtual que reflete uma porta real do hospedeiro, geralmente usando *tags* VLAN para segregar o tráfego dos contêineres. Esse *driver* entrega aos contêineres endereços de *hardware* (MAC) válidos permitindo o mais próximo de uma rede nativa para contêineres. Docker *overlay* é uma solução de comunicação de contêineres multi-hospedeiro, expande sobre o Docker *bridge* que é usado em cada hospedeiro e cria duas redes de sobreposição entre os hospedeiros: uma de controle e uma de dados, pela qual o tráfego dos contêineres é enviado.

Com exceção de um cenário específico de Docker *macvlan*, todos os *drivers* descritos limitam o acesso entrante aos serviços hospedados em contêineres à combinação de IP do hospedeiro com porta específica do(s) contêiner(es). Ainda, contêineres são usados em *data centers* multiusuários, compartilhando servidores entre diversas aplicações. Assim, o uso de contêineres requer medidas de precaução para melhorar o isolamento, como implementação de contêineres sobre máquinas virtuais e de redes virtuais privadas – redes de sobreposição com propósito de garantir isolamento e segurança nos dados trafegados entre máquinas virtuais participantes. Outra forma de conseguir o isolamento é usando *drivers* de rede para contêineres que implementam as medidas de segurança desejadas. Um exemplo é o Docker *overlay*, que pode criptografar o tráfego.

Em suma, diversas tecnologias são utilizadas para fornecer conectividade entre contêineres. Mesmo tecnologias consolidadas de virtualização de redes podem impactar o desempenho das aplicações finais. É fato que a definição de qual *driver* de rede é adequado para fornecer conectividade é uma tarefa complexa.

## 3. Trabalhos Relacionados

A literatura sobre análise de desempenho de redes para processamento de alto desempenho é vasta. Entretanto, o presente trabalho revisa àquelas diretamente relacionadas com o gerenciamento de contêineres. [Kratzke 2015] analisa o desempenho de redes de contêineres Docker com uma aplicação de testes denominada *ping-pong* na qual um cliente solicita um volume de dados para um servidor, repetindo solicitações a cada vez que recebe os dados. As métricas coletadas são de taxa de atendimento em pacotes por segundo e taxa de transferência em *bytes* por segundo.

[Morabito et al. 2015] compara o desempenho de contêineres contra virtualização convencional. A análise no quesito rede do artigo é feita sobre testes ponto-a-ponto de transferência e de requisições por segundo (similar ao *ping-pong*) usando protocolos TCP e UDP. Por sua vez, [Claassen 2015] avalia o comportamento de redes para contêineres com alta capacidade em testes de transferência com até 256 contêineres competindo pela banda disponível em cenários intra-nó e no máximo 16 pares de contêineres entre hospedeiros. Somente a taxa de transferência é comparada.

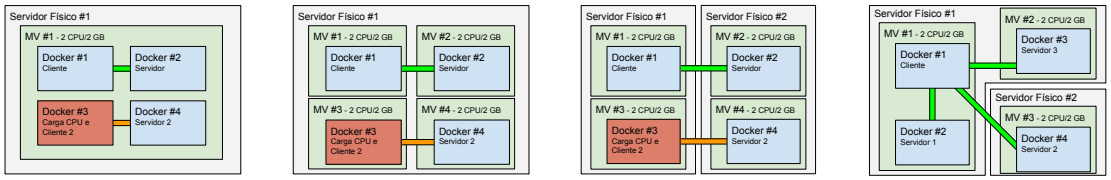
[Zismer 2016] compara quatro *drivers* de sobreposição para Docker em taxa de transferência e latência em redes ponto-a-ponto e em redes multi-salto, en-

quanto [Suo et al. 2018] põe uma coletânea de *drivers* de redes para contêineres à prova em testes de desempenho de redes para avaliar a degradação observada por executar os contêineres sobre máquinas virtuais. A taxa de transferência e latência são quantificadas. Por fim, [Jawarneh et al. 2019] compara desempenho em vários quesitos, entre virtualização baseada em KVM contra contêineres Docker e LXD. Dentre as métricas coletadas estão taxa de transferência UDP e TCP em testes de saturação de banda e latência.

Analisando os trabalhos relacionados, observa-se que a coleta de taxa de transferência e de latência são padrão em estudos de desempenho de redes e que há um grande interesse em avaliar redes de sobreposição para contêineres. Entretanto, é pouco estudado o comportamento das redes para contêineres em cenários e testes que mais aproximem as condições às quais contêineres são sujeitos em aplicações de mundo real.

## 4. Proposta e Perspectivas

Neste trabalho, é proposto uma coleção de testes para avaliação de desempenho de redes de contêineres comparando os *drivers* que acompanham uma instalação padrão do Docker. Os *drivers* avaliados serão *host* – tomado como linha de base de desempenho –, *bridge*, *macvlan* e *overlay*. São planejados três testes para coleta de métricas. O primeiro teste é de utilização de banda com a ferramenta iPerf3 [iPerf 2019]. O segundo teste usa a ferramenta SockPerf [Mellanox 2019] para medição de latência total do circuito (*Round-Trip Time*). O terceiro teste visa replicar uma curva de distribuição que representa o tráfego de data-centers disponível no artigo de [Kandula et al. 2009] com a ferramenta Empirical Traffic Generator [Cisco 2019] e fazendo a medição de taxa de transferência e tempo de conclusão de fluxo.




(a) Contêineres em uma só MV. (b) Contêineres isolados no mesmo hospedeiro. (c) Contêineres isolados em dois hospedeiros. (d) Representação de *data center*.

**Figura 1. Configurações usadas para os testes. Linha verde representa o tráfego avaliado e em laranja o tráfego de fundo, quando presente.**

Para comparação, quatro cenários serão investigados: Referência – somente os dois contêineres dos testes são executados; Interferência – um terceiro contêiner satura o processador com testes de estresse usando a ferramenta stress-ng [King 2019]; Concorrência – além dos contêineres executando os testes mais dois contêineres trafegam com iPerf3 na mesma rede para medir o comportamento de concorrência dos *drivers*; Simulação – quatro contêineres participam da simulação de tráfego de data-center com o Empirical Traffic Generator. Um contêiner é o cliente e solicita tráfego para os três servidores que entregam como resposta.

Os cenários serão executados em quatro configurações usando até quatro máquinas virtuais em duas máquinas físicas do Laboratório de Processamento Paralelo e Distribuído (LabP2D). A Figura 1 apresenta as configurações utilizadas para os testes. As três primeiras configurações serão usadas para repetir os cenários referência,

interferência e concorrência com variações no caminho que o tráfego avaliado percorre, enquanto a quarta configuração será usada exclusivamente para o teste simulação.

## 5. Conclusão

É observado o interesse no uso de redes de sobreposição para contêineres e portanto o interesse em estudar diferenças de desempenho entre *network drivers*. A proposta de testes deste trabalho aborda tanto cenários controlados quanto simulação de *data center* com métricas coerentes e trará informações importantes no desenvolvimento de trabalhos futuros aos quais este artigo fundamenta.

**Agradecimentos**: LabP2D, UDESC e FAPESC.

## Referências

Cisco (2019). datacenter/empirical-traffic-gen: Simple client-server application for generating user-defined traffic patterns. https://github.com/datacenter/empirical-traffic-gen.

Claassen, J. (2015). Container network solutions research project 2. Research paper, *FNWI / Instituut voor Informatica*, University of Amsterdam. https://scripties.uba.uva.nl/search?id=588700.

Docker, Inc. (2019). Enterprise container platform - docker. https://docker.com/.

iPerf (2019). iPerf. https://iperf.fr/.

Jawarneh, I. M. A., Bellavista, P., Foschini, L., Martuscelli, G., Montanari, R., Palopoli, A., and Bosi, F. (2019). Qos and performance metrics for container-based virtualization in cloud environments. In *Proceedings of the 20th International Conference on Distributed Computing and Networking*, pages 178–182. ACM.

Kandula, S., Sengupta, S., Greenberg, A., Patel, P., and Chaiken, R. (2009). The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 202–208. ACM.

King, C. I. (2019). stress-ng. https://kernel.ubuntu.com/~cking/stress-ng/.

Kratzke, N. (2015). About microservices, containers and their underestimated impact on network performance. *CLOUD COMPUTING 2015*, pages 165–169.

Mellanox (2019). Mellanox/sockperf: Network benchmarking utility. https://github.com/Mellanox/sockperf.

Morabito, R., Kjällman, J., and Komu, M. (2015). Hypervisors vs. lightweight virtualization: a performance comparison. In *2015 IEEE International Conference on Cloud Engineering*, pages 386–393. IEEE.

Suo, K., Zhao, Y., Chen, W., and Rao, J. (2018). An analysis and empirical study of container networks. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 189–197. IEEE.

Vaughan-Nichols, S. J. (2006). New approach to virtualization is a lightweight. *Computer*, 39(11):12–14.

Zismer, A. (2016). Performance of docker overlay networks. Research paper, *FNWI / Instituut voor Informatica*, University of Amsterdam. https://scripties.uba.uva.nl/search?id=621172.