Virtualização – Análise de Desempenho

Gabriel Sperb Stoffel[1], **Orlando Rodrigues Junior**[1], **Valmir Palacio Freitas Junior**[1]

[1]Curso de Bacharelado em Ciência da Computação – Universidade do Vale do Rio dos Sinos (UNISINOS) – Campus de São Leopoldo

Caixa Postal 950 – 93.020-190 - São Leopoldo – RS – Brasil

{gstoffel, orlandohj, valmirfj}@edu.unisinos.br

Abstract: *This article approaches a brief introduction to each technology and its main characteristics, evolutions for each virtual machines and containers, comparing them regarding their predecessor performance based on I/O Bound and CPU Bound tests, costs minimization for deployment and ease of use.*

Resumo: *Este artigo aborda uma breve introdução a cada tecnologia e suas principais características, as evoluções para máquinas virtuais e containers comparando-os em relação a desempenho de seu antecessor com base em testes de entrada/saída predominante e processamento predominante, minimização de custo para implementação e facilidade de uso.*

1. Introdução

Antes da virtualização e conteinerização era necessário ter diversos servidores para conseguir executar aplicações que por muitas vezes utilizavam recursos limitados ou que rodavam em sistema operacionais diferentes umas das outras, essa administração de servidores gerava custo por manter tantos servidores ativos e da alta complexidade que era realizar comunicação entre aplicações executando hardwares distintos.

Para então melhorar a situação surgiu a virtualização onde levou o conceito de ser possível ter apenas um único servidor com um sistema operacional e acima dele uma camada de aplicação que tem por dever dividir, isolar e gerenciar recursos do servidor para outros sistemas operacionais que ficaram acima desta camada de aplicação, cada um com suas próprias aplicações.

Passado algum tempo houve outra grande evolução onde foi percebido que por muitas vezes ter que subir uma nova máquina virtual com todo um novo sistema operacional para sustentar apenas uma aplicação era desperdício de recurso do hardware hospedeiro, com isto houve uma evolução para o que hoje conhecemos como containers e será desta evolução e entendimento da melhoria que será apresentada durante o documento.

2.Virtualização

A virtualização funciona como um intermediário entre os sistemas convidados, emulando um hardware virtual a partir de recursos disponibilizados pelo sistema operacional hospedeiro que por sua vez tem contato com o hardware e fornece acesso indireto a ele, onde a aplicação deve controlar e isolar o acesso de todos os sistemas convidados ao hardware real. Estes sistemas operam de forma isolada, ou seja, não compartilham informações entre si e nem fazem comunicação direta com camadas

inferiores, não podendo acessar diretamente o hardware ou o sistema operacional que está executando o virtualizador. [UHLIG et al, 2005].

Um dos principais motivos para utilizar a virtualização é a possibilidade de se obter várias máquinas virtuais, executando múltiplas tarefas em paralelo, dentro de uma única máquina física, o que muitas empresas optam, por ser uma opção mais barata, econômica na questão de espaço físico e possibilita dedicar uma máquina virtual por cliente ou serviço específico, além de reduzir a chance de problemas de hardware já que ocorre a redução de maquinas reais em funcionamento. [MENASCÉ, 2005].

3. Container

Container é um nível de virtualização que possibilita a execução de múltiplas aplicações isoladamente, necessitando de um host que seja compatível com tecnologia de containers, como algumas distribuições do Linux na qual o *Kernel* deve estar em uma versão igual ou posterior a 3.8 e em algumas versões do Windows. Os contêineres se comunicam diretamente com o *kernel* do sistema operacional hospedeiro, mas possuem seu próprio sistema de arquivos [SCHEEPERS 2014].

O container acabou se destacando por causa do nível de segurança em que foi possível atingir e também pelo fato de não necessitar de uma camada de sistema operacional para cada aplicação, diferente da virtualização. Outros pontos em que os containers se destacaram foram pela menor demanda por recursos, consumindo menos espaço em disco e sua portabilidade que antes por outras plataformas era algo mais difícil [JOY 2015].

4. Experimento

Será demonstrado através de experimento de CPU *bound* e I/O *bound* a comparação entre container e máquina virtual, e para tal utilizou de recursos de sistema operacional hospedeiro Ubuntu 17.04 com o *kernel* 4.10.0-38-generic sendo executado em uma CPU Intel® Core™ i7-4500U de 4 núcleos e 1.80GHz cada, memória RAM de 7,7 GiB e HD de 7200RPM e 64MB de cache para ambos os casos. Em relação ao sistema operacional hóspede utilizou-se as mesmas especificações sem delimitar o hardware. A versão do KVM utilizado foi QEMU *emulator version* 2.8.0 (Debian 1:2.8+dfsg-3ubuntu2.7) e o Docker na versão 17.10.0-ce, build f4ffd25.

Ambas as tecnologias executaram através do mesmo sistema operacional hospedeiro e foram responsáveis por subir e executar a aplicação desenvolvida por própria autoria na linguagem C++, no qual realizou o carregamento do conteúdo de um arquivo e após salvou os dados de forma ordenada em cinquenta arquivos que foram criados posteriormente pelo mesmo.

Para acompanhar a carga de I/O *Bound* da máquina utilizou-se o comando *iostat -x -t* disponível no pacote sysstat 11.5.7-1ubuntu1 do repositório oficial, enquanto para identificar a carga de CPU *Bound* utilizou-se o comando *top -p [pid]*.

4.1. I/O Bound

Durante a execução do experimento notou-se na Figura 1 que no caso do Docker há uma quantidade constante de escrita, fazendo com que seja mais eficiente em relação ao

KVM que tem altos picos de escrita, assim exigindo que o disco suba suas rotações em um curto período de tempo. Essas nuances agudas fazem com que o mesmo perca performance e faça com que vários dados não sejam salvos caso ocorra algum problema durante os estados letargia.

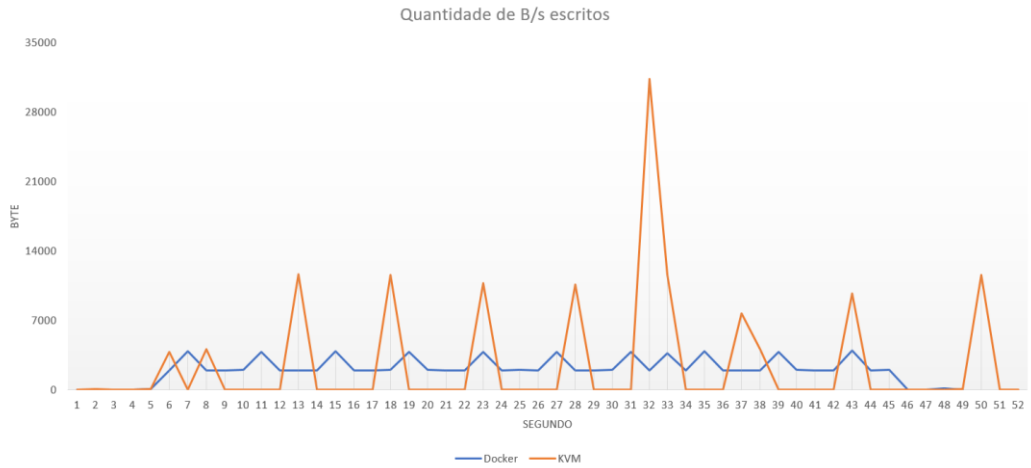

Figura 1. Gráfico de B/s escritos. Fonte: Autoria própria.

Seguindo com a Figura 2 afirmou-se o que está sendo exposto na Figura 1 acima onde o Docker conteve uma maior quantidade de requisições de leitura fazendo com que a quantidade de B/s seja mais contínua e estável, enquanto isto afirmou-se que no KVM a situação é exatamente oposta, onde há menor quantidade de requisições, geram picos maiores de B/s, já que em apenas uma requisição ele deve fornecer uma grande quantidade de dados.

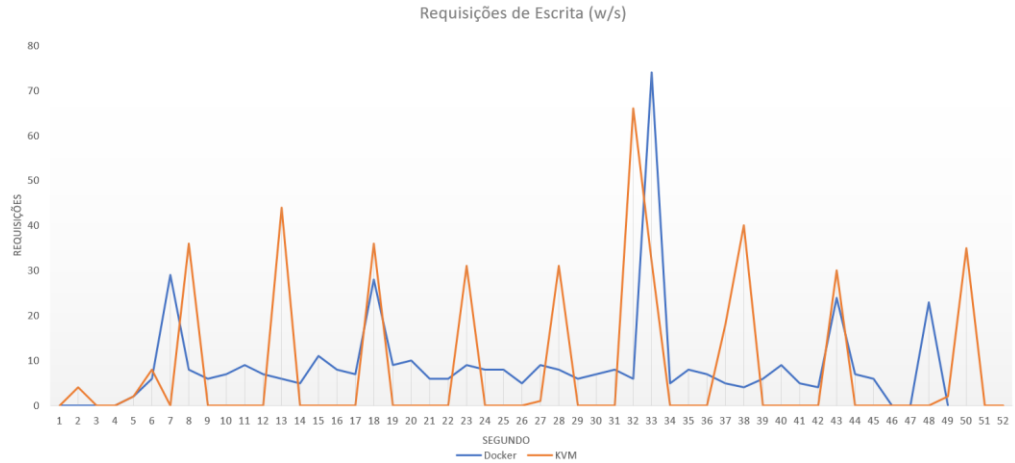

Figura 2. Gráfico de Requisições de Escrita (w/s). Fonte: Autoria própria.

4.2. CPU Bound

Para a análise de CPU verificou-se individualmente o quanto utilizou-se em modo usuário e modo *kernel* para não haver distorção de valores na comparação. A partir disto a Figura 3 e Figura 4 demonstra o KVM com um maior percentual de uso de CPU em ambos o caso, resultando em uma diminuição de desempenho e um desgaste maior para a máquina.

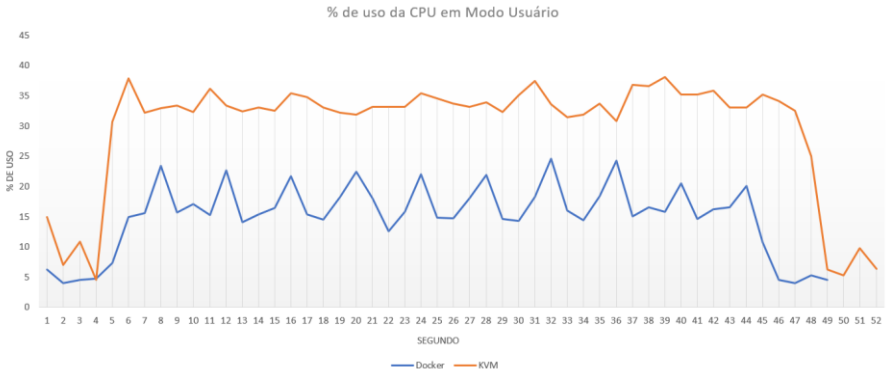

Figura 3. Gráfico de uso da CPU em modo usuário. Fonte: Autoria própria.

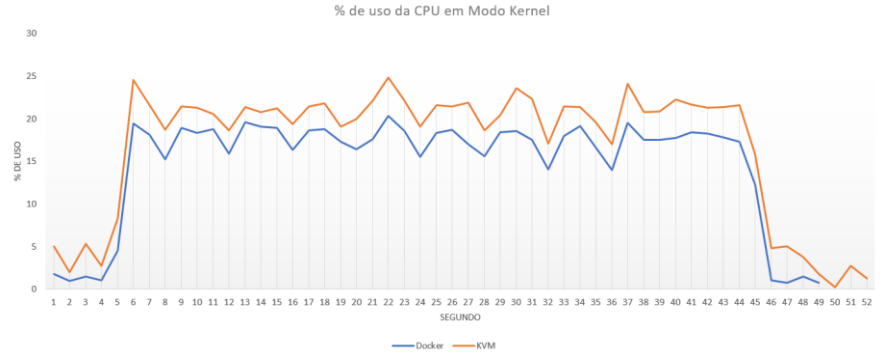

Figura 4. Gráfico de uso da CPU em modo *kernel*. Fonte: Autoria própria.

5. Conclusão

A partir do estudo de referencial teórico, foi possível destacar as diferenças de implementação entre duas técnicas de virtualização embora ambas compartilham objetivos similares.

Tendo em vista os resultados apurados com a execução dos testes, pode-se destacar o contêiner como uma ferramenta de maior performance para este caso de estudo sobre I/O e CPU bound, dado que a conclusão de execução terminou de forma mais rápida, estável e consumindo uma quantidade menor de recursos da máquina.

Referências

JOY, Ann Mary (2015). “Performance comparison between linux containers and virtual machines. In: Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in. IEEE”, p. 342-346.

MAUERER, Wolfgang. “Professional Linux® Kernel Architecture”. 2008.

MENASCÉ, Daniel A. (2005). “Virtualization: Concepts, Applications and Performance Modeling".

SCHEEPERS, Mathijs Jeroen (2014).”Virtualization and containerization of application infrastructure: A comparison. In: 21st Twente Student Conference on IT“. p. 1-7.

UHLIG, Rich; NEIGER, Gil; RODGERS, Dion; SANTONI. Intel Virtualization Technology. Computer: Volume 38, Issue 5. 2005. p. 48-56.