Análise de desempenho e segurança dos orquestradores Apache Mesos, Docker Swarm e Kubernetes em nuvens baseadas em OpenStack

Kerolayne de Souza Vieira de Oliveira[1], **Charles Christian Miers**[1]

[1] Departamento de Ciência da Computação (DCC)
Universidade do Estado de Santa Catarina (UDESC)

kerolayne.oliveira@edu.udesc.br, charles.miers@udesc.br

Resumo. *O uso de orquestradores é tipicamente empregado para realizar o gerenciamento e organização dos contêineres tanto do seu ciclo de vida como da sua interconexão em rede. O objetivo deste artigo é apresentar uma proposta para analisar o desempenho e segurança disponibilizados pelos orquestradores Apache Mesos, Kubernetes e Docker Swarm em nuvens computacionais baseadas em OpenStack.*

1. Introdução

As nuvens computacionais baseadas em OpenStack são a opção preferida para implantar nuvens *Infrastructure-as-a-Service* (IaaS) privadas baseadas em software livre. Na primeira década de sua existência, o OpenStack consolidou-se como solução IaaS oferecendo virtualização através de máquinas virtuais (MVs) em uma considerável plataforma de hipervisores. A disponibilidade de usar contêineres na plataforma OpenStack possui diversas formas de implementação, assim como maneiras distintas de gerenciamento. Neste contexto, os contêineres entram são considerados uma solução atraente por possuírem implantação com poucos recursos computacionais e proporcionarem um ambiente básico. Porém, para sistemas mais complexos que necessitam de milhares de microsserviços isso pode resultar em problemas de uma ordem expressiva de manutenção, com isso os orquestradores agem como facilitadores na implantação e gestão destes contêineres [Jawarneh et al. 2019].

O OpenStack disponibiliza o serviço Magnum como uma das opções para gerenciar contêineres. O Magnum possui compatibilidade nativa com as soluções de orquestração de contêineres Apache Mesos, Docker Swarm e Kubernetes [OpenStack.org 2020d]. O objetivo deste artigo é apresentar uma proposta para analisar o desempenho e segurança disponibilizados pelos orquestradores Apache Mesos, Kubernetes e Docker Swarm em nuvens OpenStack.

2. OpenStack

O OpenStack é um software desenvolvido para criar nuvens IaaS tanto públicas como privadas, proporcionando controle dos recursos de computação, armazenamento e rede através de uma *Application Programming Interface* (API) [OpenStack.org 2020a]. Este disponibiliza uma variada gama de aplicações para auxiliar na gestão de uma nuvem, e entre estas estão: o Magnum (Figura 1) e o Zum que possibilitam o uso de contêineres nativamente no OpenStack, ainda é possível também utilizar Nova-Docker que é um *driver* virtual [OpenStack.org 2020c].

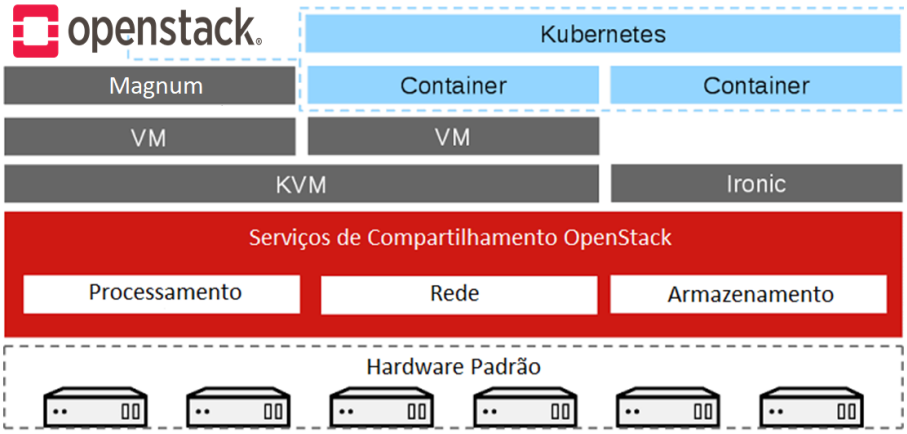

Figura 1. Diferentes formas de operar contêineres no OpenStack.

O Magnum possui a principal API de contêineres do OpenStack, possibilitando o uso de *Container Orchestrator Environment* (COE) como ferramentas nativas. o OpenStack utiliza o serviço Heat para orquestrar a imagem do sistema operacional (SO) que possui o Docker, sendo que o COE replica essa imagem em configuração de *cluster*, indiferente se em MV ou em *bare metal* [OpenStack.org 2020b]. Tal fato permite que múltiplos *clusters* executem simultaneamente enquanto usam diferentes orquestradores [OpenStack.org 2020d].

3. Orquestradores

Os orquestradores atuam como facilitadores na gestão de *clusters* de contêineres e são compostos por três camadas básicas que são [Jawarneh et al. 2019]:

- Gerenciamento de recursos: gerencia recursos de baixo nível. Por exemplo: espaço em disco, volume, CPU/GPU.
- Agendamento: visa o uso eficiente dos recursos do *cluster*. Exemplo: replicação e dimensionamento do número de réplicas.
- Gerenciamento de serviços: disponibiliza recursos para possibilitar criar e implantar aplicativos corporativos, *e.g.*, *cgroups* e *namespaces* ou dependência entre microsserviços.

As características relevantes dos orquestradores abordados estão listadas na Tabela 1.

Tabela 1. Principais características dos orquestradores.

Orquestradores	Escalabilidade	*Deploy* de contêineres
Kubernetes	O componente *Replication Controller* efetua a ação.	Os *pods* são responsáveis pelo controle e distribuição dos contêineres.
Docker Swarm	Além da escalabilidade padrão de contêineres, permite a escalabilidade de serviços.	Permite implantar varias imagens a partir de uma única imagem já utilizada no Docker através dos *slaves*.
Apache Mesos	Kernel escalável e resistente para permitir que estruturas compartilhem *clusters*.	os *masters* implementam o compartilhamento entre estruturas, e lista os recursos disponíveis nos *slaves*.

4. Proposta: Segurança e Desempenho

Em avaliações anteriores, com a abordagem focada apenas em segurança de contêineres [Oliveira and Miers 2019], pôde-se chegar aos resultados listados na Tabela 2.

Tabela 2. Resultados da análise de segurança [Oliveira and Miers 2019].

Critérios		Vulnerabilidades	Possíveis soluções	Boas práticas
Controle e limitação de recursos.	Má configuração do *AppArmor*.	Escalação de privilégios.	Verificar se o mecanismo está ativado; Utilizar o profile padrão oferecido pelo Docker.	Auditar as políticas de segurança baseado no contexto necessário para a aplicação.
	Permissão indevida de capacidades do núcleo.	Negação de serviço; Execução de código arbitrário.	Utilizar apenas as capacidades do profile padrão do contêiner.	Remover as capacidades que não serão utilizadas no contexto da aplicação; Verificar a possibilidade remover a CAP_SYS_ADMIN, responsável pela concessão da maior parte dos privilégios de super usuário.
	Não filtragem de chamadas Seccomp.	Escalação de privilégios.	Utilizar o profile padrão do Docker com a permissão de entorno de 311 chamadas de sistema.	Não modificar o profile padrão.
Comunicação entre contêineres.		ARP Spoofing, *Man-in-the-Middle.*	Cifragem do canal de comunicação; Criação de redes definidas pelo usuário.	Não utilizar a interface *docker0*; Uso de serviços de orquestração (e.g., Docker Swarm).

Em relação ao desempenho algumas literaturas apontam parâmetros comuns entre os orquestradores que podem ser usados para a comparação de desempenho [Watada et al. 2019, Jawarneh et al. 2019]. A pesquisa de desempenho é nova fase em execução da pesquisa, que acopla-se a segurança no sentido de auxiliar nos aspectos relacionados a negação de serviço (DoS). Com base na pesquisa realizada, foram selecionados os seguintes critérios de análise:

- Análise do tempo de implantação de *cluster*.
- Tempo de *Failover*.

Para análise destes critérios foi realizada a instrumentação de um ambiente de testes na nuvem OpenStack do LabP2D/UDESC, a Nuvem Tche. Foram criados ambientes de contêineres com os orquestradores usando como base contêineres Docker, na versão Community em máquinas virtuais com o sistema operacional GNU/Linux Ubuntu. A ferramenta *lifecycle management*, nativa do arcabouço Docker, é empregada para auxiliar no gerenciamento dos contêineres (usando um ambiente Nova-Docker).

Os resultados de experimentos iniciais revelaram que o Kubernetes teve um melhor resultado, com exceção no teste de *Failover* no qual a falha de um único contêiner é rapidamente resolvida, diferente de quando a falha ocorre em um nó, neste caso o Kubernetes teve o pior resultado. O Apache Mesos também apresentou pontos positivos como no teste de tempo de implantação do *cluster*, além e conseguir otimizar seus resultados no teste de implantação de uma aplicação da *web*. O Docker Swarm conseguiu fornecer o menor tempo de *Failover*.

5. Considerações e Trabalhos futuros

A análise referente a segurança apontou os orquestradores como mais uma barreira efetiva para a proteção do funcionamento dos contêineres, com isso o uso correto das mesmas já

é uma boa prática. A mesma abordagem em ambiente Magnum, neste passo ainda será levantada uma possível ferramenta de *benchmark* e como serão efetuados os testes.

Agradecimentos: Os autores agradecem o apoio do LabP2D/UDESC e a FAPESC.

Referências

Jawarneh, I. M. A., Bellavista, P., Bosi, F., Foschini, L., Martuscelli, G., Montanari, R., and Palopoli, A. (2019). Container orchestration engines: A thorough functional and performance comparison. In *IEEE ICC 2019*, pages 1–6.

Oliveira, K. and Miers, C. (2019). Análise de segurança no uso de contêineres Docker em máquinas virtuais. In *17a Escola Regional de Redes de Computadores*, Alegrete-RS, Brasil.

OpenStack.org (2020a). Construa o futuro da infraestrutura aberta. https://www.openstack.org/.

OpenStack.org (2020b). Magnum - OpenStack. https://wiki.openstack.org/wiki/Magnum.

OpenStack.org (2020c). O que é o OpenStack? https://www.openstack.org/software/project-navigator/openstack-components#bridges-for-adjacent-tech.

OpenStack.org (2020d). OpenStack docs: Magnum guia do usuário. https://docs.openstack.org/magnum/latest/user/#choosing-a-coe.

Watada, J., Roy, A., Kadikar, R., Pham, H., and Xu, B. (2019). Emerging trends, techniques and open issues of containerization: A review. 7:152443–152472.