Elasticidade Vertical de Memória em Docker Containers*

Daniel M. B. Sodré, José Victor de P. e Silva, Cristina Boeres, Vinod E. F. Rebello

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brazil

{danielbougleux,josevictorsilva}@id.uff.br,{boeres,vinod}@ic.uff.br

Resumo. *Nuvens computacionais executam as requisições de seus clientes em um ambiente virtual configurado de acordo com as necessidades das aplicações. O cliente normalmente paga por uma quantidade de recursos estimada para sua aplicação executar, mas esta pode vir a consumir mais ou menos recursos do que o esperado. Para evitar tais situações, o gerenciamento eficiente de recursos é essencial. Esse trabalho visa elucidar alguns pontos estudados sobre o consumo de recursos de aplicações executando em Docker containers.*

1. Introdução

Plataformas de computação em nuvem visam uma melhor utilização de seus recursos para poder melhor atender os seus clientes. É comum, no entanto, encontrar aplicações executando na nuvem com subutilização dos recursos alocados. Neste caso, tais recursos poderiam ser associados a outras aplicações. Por outro lado, pode ocorrer o cenário oposto, onde a aplicação consome mais recursos do que o estimado, o que pode levar a uma degradação de seu desempenho ou até mesmo o término precoce.

É interessante, então, que provedores de serviços em nuvem sejam capazes de escalar os recursos da nuvem para atender a uma variação na demanda de maneira automática – essa habilidade chama-se elasticidade de recursos. Há dois tipos de elasticidade: elasticidade vertical, onde o sistema é capaz de prover mais ou menos recursos computacionais baseado na demanda das aplicações em uma instância computacional; e elasticidade horizontal, onde o sistema é capaz de aumentar ou diminuir suas instâncias computacionais para atender a variações na demanda.

O *Vertical Elasticity Management of Containers* (VEMoC) [Nicodemus et al. 2020] através da análise e previsão do consumo de memória das aplicações, implementa elasticidade vertical de memória em conjunto com políticas de preempção dos contêineres levando a resultados promissores tanto para o provedor quanto para os seus clientes. Considerando que os mecanismos do VEMoC utilizam LXC [Canonical Ltd. 2020], este trabalho tem por objetivo estender o framework para outras tecnologias de conteineirização como Docker[Docker Inc. 2020], uma plataforma bastante utilizada. Experimentos iniciais do Docker apresentaram certa sobrecarga de monitoramento e suspensão de contêineres. Por conta disso, uma análise mais aprofundada de alternativas para monitoramento de contêineres e mecanismos de checkpoint se faz necessária, para integrar Docker ao VEMoC.

*Este trabalho foi apoiado pelo projeto REMATCH (Processo nº 88887.310261/2018-00) do Programa Institucional de Internacionalização (PrInt) da CAPES e os alunos Daniel M. B. Sodré e José Victor de P. e Silva por bolsas de Iniciação Científica de CNPq/PIBIC 2020-21.

No caminho deste aprofundamento, este trabalho aqui apresentado mostra uma análise inicial dos benefícios e desafios de gerenciadores de recursos na nuvem para aplicações executadas em Docker contêineres. Docker conta com uma implementação do Kernel Linux chamada *control groups* para gerenciar os recursos do contêiner, permitindo o monitoramento e a elasticidade dos recursos em ambientes Docker.

2. Elasticidade Vertical em Docker

Pode-se apontar certas diferenças na implementação de elasticidade vertical entre contêineres e Máquinas Virtuais (MVs). Os contêineres trabalham com limitação de recursos, que contrasta com a alocação efetiva dos recursos no caso das MVs. Assim, ao gerenciar o consumo de recursos de contêineres, a quantidade de recursos efetivamente alocada pelas aplicações dentro do contêiner em um determinado momento pode ser menor ou igual ao dado limite. A consequência disso é que, aquela quantidade de recursos não alocada em um determinado momento pode ser utilizada para outros processos/contêineres executados na mesma máquina. Desta forma, pode haver uma certa interferência entre recursos alocados a diferentes contêineres/processos. Assim, para gerenciar efetivamente utilização de recursos da nuvem, um *middleware* de gerenciamento deve ser capaz de manipular dinamicamente os limites de utilização de recursos dos contêineres em tempo de execução e consequentemente, melhorar o aproveitamento dos mesmos, sem causar interferência maléfica entre eles – esse é um dos conceitos implementados no VEMoC, adicionado à manipulação de eventos de preempção de contêineres para um melhor desempenho do ambiente virtual.

Seja uma aplicação submetida, onde o provedor configura o limite de memória do contêiner para M unidades, de acordo com a requisição do cliente. Duas situações em relação à quantidade de memória podem ocorrer: (a) *superestimação*: a aplicação precisa de bem menos que M de memória, levando a subutilização dos recursos; (b) *subestimação*: a aplicação precisa mais do que M de memória, levando a uma queda de desempenho da execução ou mesmo a interrupção da mesma[Kubernetes 2020].

Para monitorar o uso de memória, o VEMoC coleta informações diretamente do *cgroups*. Já o Docker, além de dados do *cgroups*, coleta um conjunto de informações que podem não ser relevantes para o problema em questão. Não somente é importante avaliar a sobrecarga associada à coleta, mas também a frequência de atualização. Avaliações iniciais mostram que a coleta do uso de memória de um contêiner usando *cgroups* é duas ordens de grandeza mais rápida do que usando o API do Docker.

3. Análise dos limites de memória

Para analisar o efeito do limite de memória na execução de aplicações em Docker contêineres, foram realizados experimentos considerando dois *jobs memory bound* sintéticos, denotados por $J1$ e $J2$. Cada um dos *jobs* acessa N elementos de um vetor. Enquanto $J1$ acessa todos os elementos em sequência em I iterações, no *job* $J2$, o vetor é dividido em m blocos tal que os elementos de cada bloco são acessados em sequência I vezes antes de passar para o próximo bloco.

Os experimentos consideraram um vetor de 4096 MB, que é percorrido $I = 4$ vezes e, no caso do $J2$, $m = 4$ blocos de 1024MB. O objetivo dos testes é avaliar o comportamento de ambas as aplicações quando executadas sob limites M de memória

variados. A versão do Docker utilizada foi a 19.03.13, com contêiner com imagem base do CentOS:7, em um máquina física com processador AMD FX(tm)-8300, 8 cores em 3.3GHz, 8GB de RAM, 1666MHz DDR3 e 12GB de swap.

A Figura 1 mostra a média dos tempos de dez execuções de cada um dos *jobs*, com $M = 512$ até 4096 MB com incrementos de 512 MB. Foram adicionados as configurações de $M = 1040$ e 4110 MB para uma análise mais detalhada. Os resultados mostram que, enquanto em aplicações como o $J1$, uma má previsão da necessidade de memória pode causar uma grande degradação em seu desempenho devido principalmente ao uso de swap pelos contêineres, manipular o limite de memória em contêineres que executam jobs como $J2$ podem levar a uma melhor utilização da memória com uma sobrecarga mínima.

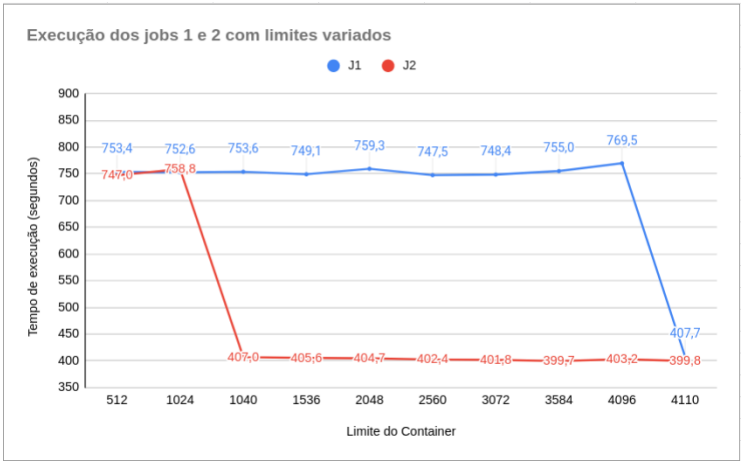

Figura 1. Execução de $J1$ e $J2$ variando limites de memória de seus contêineres.

4. Conclusão e objetivos futuros

A análise cuidadosa do uso de recursos em um ambiente virtual com Docker contêineres pode levar a um melhor custo-benefício na utilização de nuvens, tanto por parte dos usuários quanto dos provedores. Futuramente esse trabalho se desenvolverá para viabilizar a utilização de Docker contêineres no VEMoC, um gerenciador que explora elasticidade vertical em um ambiente com contêineres [Nicodemus et al. 2020], que atualmente é viável apenas para contêineres LXC. Isso será realizado através do aprofundamento dos desafios que o Docker traz na implementação da ferramenta e em como resolvê-los.

Referências

Canonical Ltd. (2020). Linux containers - lxc. `https://linuxcontainers.org/pt_br/lxc/introduction`. Accessed on: 27/02/2020.

Docker Inc. (2020). Docker get started. `https://docs.docker.com/get-started/`. Accessed on: 10/07/2020.

Kubernetes (2020). Kubernetes. `https://kubernetes.io/`. Accessed on: 01/07/2020.

Nicodemus, C., Boeres, C., and Rebello, V. E. F. (2020). Managing vertical memory elasticity in containers. In *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pages 1–10.