Migração de Contêineres Utilizando runC e CRIU em Ambiente de Nuvem

Matheus V. R. Nascimento[1], **Daniel Cordeiro**[1]

[1]Escola de Artes, Ciências e Humanidades – Universidade de São Paulo (USP)
São Paulo – SP – Brasil

{mvr.nascimento,daniel.cordeiro}@usp.br

Este trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), procs. #2019/26702-8, #2021/06867-2 e #2023/00811-0, do CCD Cidades Carbono Neutro (FAPESP #2024/01115-0), do CNPq (proc. #444766/2024-3) e do grupo de pesquisa THUS (Techno-Human Systems of the Future) do Centro Internacional de Pesquisa (IRC) 'Transitions' (CNRS, USP e FAPESP CIP #2025/01171-0).

Abstract. *This article investigates container migration in clouds, using runC and CRIU for transferring running/executing applications. The research details the environment setup, including all the tools used. Two applications were utilized: one to demonstrate CRIU's preservation of active TCP connections, and the other to evidence the influence of data volume on migration time. The conclusion emphasizes the feasibility of container migration to optimize resources and reduce energy consumption in data centers.*

Resumo. *Este artigo investiga a migração de contêineres em nuvens, utilizando runC e CRIU para transferências de aplicações em execução. A pesquisa detalha a configuração do ambiente, incluindo todas as ferramentas utilizadas. Foram utilizadas duas aplicações, uma para demonstrar a capacidade do CRIU em preservar conexões TCP ativas e a outra evidenciar a influência do volume de dados no tempo de migração. A conclusão ressalta a viabilidade da migração de contêineres para otimizar recursos e reduzir o consumo de energia em data centers.*

1. Introdução

A computação em nuvem revolucionou a forma como as aplicações são desenvolvidas e implementadas, oferecendo escalabilidade, flexibilidade e eficiência nos custos. No entanto, o crescimento exponencial da demanda por recursos computacionais em nuvem trouxe consigo desafios, especialmente no que diz respeito ao consumo de energia e à otimização da utilização dos recursos. A estrutura física na computação em nuvem é composta por *data centers* (DCs), e sua crescente demanda por energia elétrica intensifica o desafio. Atualmente, os DCs demandam cerca de 2% do consumo global de eletricidade[1], e esse número tende a aumentar consideravelmente com a expansão do setor, impulsionado pela inteligência artificial e pela digitalização da sociedade.

[1]International Energy Agency, Electricity 2024: https://www.iea.org/reports/electricity-2024

Neste contexto, a consolidação de recursos, agrupando várias cargas de trabalho em um número menor de máquinas físicas, otimiza a utilização de recursos e reduz o consumo de energia [Piraghaj et al. 2015]. A utilização de contêineres reduz a sobrecarga necessária para realizar a migração, reduz o tempo de inatividade e o número de páginas de memória transmitidas [Bhardwaj et al. 2024], reduzindo também o impacto na infraestrutura de rede. Este trabalho explora a migração de contêineres utilizando ferramentas como o runC [Espe et al. 2020], um *runtime* de contêineres leve e eficiente para executar de forma isolada a aplicação, e o CRIU [2], uma ferramenta de *checkpoint*/*restore* de processos amplamente estudada e utilizada na comunidade acadêmica, no ambiente de nuvem, detalhando o processo de implementação.

2. Contêineres e Virtualização

A conteinerização é uma tecnologia que permite empacotar e isolar uma aplicação com todo o seu ambiente de execução. Estes pacotes são chamados de contêineres e permitem que as aplicações sejam executadas de forma rápida e confiável entre diferentes ambientes [Silberschatz et al. 2018]. Diferentemente de máquinas virtuais (VMs) que virtualizam o *hardware* através de um hipervisor, os contêineres utilizam recursos do sistema operacional (SO), compartilhando o *kernel* do hospedeiro, resultando em uma menor sobrecarga, maior eficiência e maior portabilidade [Morabito et al. 2015].

3. runC e CRIU

O runC é uma ferramenta de baixo nível com interface de linha de comando para execução de contêineres em conformidade com as especificações do *Open Container Initiative* (OCI) cuja principal função é criar e executar contêineres. Sua implementação utiliza funcionalidades nativas do *kernel*, como *Namespaces* e *CGroup*, para criar ambientes de execução isolados entre si e entre o hospedeiro. O CRIU (*Checkpoint/Restore in Userspace*) é uma ferramenta que permite congelar o estado atual de uma árvore de processos e salvá-la em imagens que podem ser usadas para restaurar todos os processos que foram congelados. Essa capacidade é fundamental para a migração, pois possibilita a transferência das aplicações e posterior restauração em outro hospedeiro. O CRIU opera no espaço do usuário, sem a necessidade de modificações do *kernel*.

4. Experimentos

Esta seção detalha os experimentos conduzidos para avaliar a viabilidade da migração transparente para a aplicação de contêineres com conexões TCP ativas. O objetivo principal é verificar a capacidade das ferramentas de migrar de forma transparente os contêineres entre VMs e quantificar o tempo necessário para realizar as migrações sob diferentes tamanhos de dados em memória no contêiner.

4.1. Ambiente dos Experimentos

O ambiente para a realização dos experimentos foi configurado utilizando a nuvem pública do Google, com três instâncias virtuais na mesma região com as seguintes especificações: 2 vCPU, 1GB de memória RAM e 10GB de espaço em disco. A Figura 1 ilustra o ambiente de execução dos contêineres, detalhando os endereços IP das

[2] `https://criu.org/Main_Page`

VMs e o endereço do *namespace* de rede dos contêineres, necessários para estabelecer a conexão durante os experimentos. A distribuição do SO Linux utilizada foi o Debian 12 com a versão do Linux *kernel* 6.1.0-31. A versão das demais ferramentas utilizadas inclui: **runC** (1.3.0-rc.1), **CRIU** (4.1), **GO** (1.24.2), **Skopeo** (1.9.3), **umoci** (0.4.7), **Rsync** (3.2.7), **netns** (0.5.3), **Crit** (7.2.0), **redis-cli** (7.4.1), **imagem busybox** (1.37-glibc) e **imagem redis** (7-alpine). Um repositório [3] no GitHub foi criado contendo um roteiro para a preparação e execução dos experimentos.

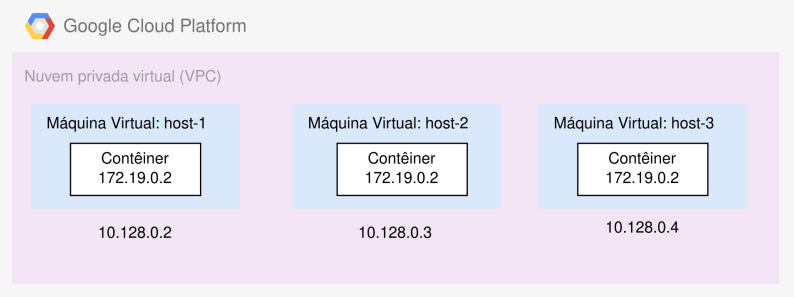

Figura 1. Diagrama do ambiente de testes

4.2. Aplicação do Experimento

Para simular um cenário de migração real, foi desenvolvida uma aplicação em linguagem C, composta por um servidor e um cliente, que estabelecem uma comunicação TCP. O servidor inicia e aguarda a conexão de um cliente. Ao receber uma conexão, o servidor ecoa todas as mensagens enviadas pelo cliente e incrementa um contador de mensagens. O cliente, por sua vez, inicia conectando-se ao servidor e envia mensagens em um loop contínuo com intervalos de 1 segundo . Essa aplicação foi escolhida por sua simplicidade e capacidade de demonstrar a migração de contêineres com conexões TCP ativas. Para simular migrações de contêineres com diferentes tamanhos, foi utilizada a aplicação Redis, com diferentes quantidades de dados carregados em memória e foi mensurado o tempo para realizar a migração.

4.3. Descrição do experimento de validação da migração com conexão TCP ativa

1. **Servidor**: Foi iniciado o servidor da aplicação em um contêiner no host-2;
2. **Cliente**: Um contêiner executando o cliente da aplicação foi iniciado no host-1;
3. **Comunicação**: Verificou-se a comunicação bem-sucedida entre os contêineres, confirmando o incremento do contador de mensagens;
4. ***Checkpoint* do Servidor**: O estado do contêiner servidor no host-2 foi congelado com CRIU, gerando os arquivos de *dump*;
5. **Transferência dos Arquivos**: Os arquivos de *dump* foram transferidos do host-2 para o host-3 utilizando a ferramenta rsync;
6. ***Restore* do Servidor**: No host-3, o contêiner foi restaurado, retomando a execução a partir do estado congelado;
7. **Ajuste do Cliente**: Como o servidor mudou de endereço IP após a migração, no host-1 foi necessário reconstruir o *socket* com os endereços de IP atualizados. Para isso, foi realizado um *checkpoint* do cliente, os arquivos de *dump* foram analisados

[3] `https://github.com/Matheusvxz/container-migration/tree/erad`

e o arquivo $files.img$ foi decodificado usando a ferramenta CRIT, foi atualizado o parâmetro $INETSK$, responsável por armazenar informações referentes aos *sockets*, modificando o endereço IP e codificado novamente. Em seguida, o cliente foi restaurado no host-1, estabelecendo a conexão com o host-3.

4.4. Descrição do experimento de verificação do tempo de migração

1. **Execução**: Um contêiner executando o servidor Redis foi iniciado no host-2;
2. **Carga de dados**: O redis-cli foi utilizada para carregar dados no host-2;
3. ***Checkpoint* do Servidor**: Um *checkpoint* do servidor foi realizado no host-2 em um sistema de arquivos temporário para otimizar a leitura e escrita.
4. **Transferência dos Arquivos**: Os arquivos de *dump* foram transferidos do host-2 para o host-3 utilizando a ferramenta rsync;
5. ***Restore* do Servidor**: O contêiner foi restaurado no host-3, retomando a execução a partir do estado congelado;

O tempo de execução das etapas 3, 4 e 5 foi medido para três cenários: Redis vazio, com 110MB e com 785MB de dados carregados. Os resultados obtidos foram: 1,378s, 2,304s e 3,426s, respectivamente.

5. Conclusão

Este artigo explorou a migração de contêineres em ambientes de nuvem, destacando a utilização das ferramentas runC e CRIU para realizar a transferência de aplicações em execução. Foi demonstrado como a combinação dessas tecnologias permite a migração de contêineres com conexões TCP ativas, ajustando o cliente para se reconectar ao servidor migrado, evidenciando a capacidade do CRIU de manter a integridade da conexão durante o processo e como o tamanho dos dados em memória influencia diretamente no tempo de migração. Como contribuição, este trabalho disponibiliza todos os códigos referentes à utilização das ferramentas.

Referências

Bhardwaj, A., Singh, A. P., Sharma, P., Abid, K., and Gupta, U. (2024). Performance evaluation of virtual machine and container-based migration technique. In Swaroop, A., Polkowski, Z., Correia, S. D., and Virdee, B., editors, *Proceedings of Data Analytics and Management*, pages 551–558, Singapore. Springer Nature Singapore.

Espe, L., Jindal, A., Podolskiy, V., and Gerndt, M. (2020). Performance evaluation of container runtimes. In *Proceedings of the 10th International Conference on Cloud Computing and Services Science - CLOSER*, pages 273–281. INSTICC, SciTePress.

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.

Piraghaj, S. F., Dastjerdi, A. V., Calheiros, R. N., and Buyya, R. (2015). A framework and algorithm for energy efficient container consolidation in cloud data centers. In *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pages 368–375.

Silberschatz, A., Galvin, P., and Gagne, G. (2018). *Operating System Concepts*. Wiley.