Explorando a Elasticidade em Containers Docker

Gustavo Rafael De Paris Chiossi
Ciência da Computação - Unioeste
Cascavel, Brasil
gustavo.chiossi@unioeste.br

Guilherme Galante
Ciência da Computação - Unioeste
Cascavel, Brasil
guilherme.galante@unioeste.br

***Abstract*—Although solutions for virtual machine elasticity exist, their application in container-based environments remains a challenge. This project aims to investigate and implement techniques for automatic resource allocation in Docker containers, and evaluate their impact on the performance and behavior of containerized applications.**

***Keywords*—Elasticity; containers; Docker.**

***Resumo*—Embora existam soluções de elasticidade para máquinas virtuais, sua aplicação em ambientes baseados em contêineres ainda é um desafio. Este projeto propõe estudar e implementar técnicas de alocação automática de recursos em contêineres Docker, avaliando o impacto no desempenho e no comportamento das aplicações conteinerizadas.**

***Palavras-chave*—Elasticidade; contêineres; Docker.**

I. Introdução

Em nuvens computacionais, a elasticidade é uma característica fundamental que permite o dimensionamento dinâmico de recursos computacionais conforme a demanda de serviços. Em cenários onde a carga de trabalho varia de forma imprevisível, essa capacidade de ajuste automático garante a manutenção do desempenho do sistema ao mesmo tempo em que otimiza os custos operacionais [1]. Esse conceito torna-se especialmente relevante em contextos modernos que utilizam contêineres Docker, uma tecnologia amplamente adotada por sua leveza, portabilidade e eficiência na execução de aplicações em ambientes dinâmicos [2]. Os contêineres Docker permitem o empacotamento e a execução isolada de aplicações, com menor sobrecarga se comparados a outras formas de virtualização.

Nesse contexto, este trabalho propõe a investigação e implementação de um mecanismo de elasticidade voltado especificamente para contêineres Docker. A proposta envolve a construção de um controlador capaz de monitorar métricas de uso de recursos de CPU e memória em tempo de execução e, a partir desses dados, tomar decisões automáticas de ajuste de recursos para os contêineres em execução. O objetivo é garantir que as aplicações mantenham um bom desempenho mesmo sob variações de carga, ao mesmo tempo que se evita o desperdício de recursos ociosos.

A solução é validada por meio da execução de aplicações, com registro e análise de métricas de desempenho com e sem a aplicação dos ajustes elásticos. Como contribuição, o trabalho pretende auxiliar no avanço de soluções de escalabilidade automatizada para contêineres, promovendo maior eficiência em arquiteturas modernas de computação em nuvem.

O restante do trabalho é organizado como segue. Na Seção 2, apresentam-se os trabalhos relacionados. Na Seção 3, apresenta-se o desenvolvimento do controlador de elasticidade. Na Seção 4, são apresentados os resultados. Por fim, a Seção 5 conclui o trabalho.

II. Trabalhos relacionados

Como trabalhos gerais sobre o assunto, destacam-se [3], [4], que exploram uma visão mais abrangente da área. Focando especificamente no escopo de elasticidade em contêineres, alguns estudos investigam técnicas de escalonamento automático, monitoramento e alocação dinâmica de recursos para aplicações conteinerizadas.

O trabalho de [5] apresenta uma abordagem voltada ao processamento de Big Data em ambientes de nuvem baseados em contêineres, propondo o uso de elasticidade vertical para otimizar a utilização de recursos. Os autores demonstram que a alocação dinâmica de CPU e memória a contêineres, de acordo com a carga de trabalho, pode reduzir desperdícios de recursos e melhorar a eficiência do sistema.

De forma complementar, [6] propõe o ELASTICDOCKER, um mecanismo de elasticidade vertical autônomo para contêineres Docker. O estudo apresenta uma arquitetura capaz de monitorar continuamente o consumo de recursos e ajustar dinamicamente os limites de CPU e memória dos contêineres sem interrupção de serviço. Os experimentos realizados demonstram que o ajuste vertical automático melhora significativamente a utilização de recursos, reduz desperdícios e mantém o desempenho da aplicação mesmo sob cargas de trabalho flutuantes.

Embora os trabalhos citados forneçam abordagens importantes para elasticidade em contêineres, este estudo propõe um controlador de elasticidade independente de orquestradores, capaz de ajustar em tempo real os limites de CPU e memória

de contêineres Docker. Além disso, permite avaliar quantitativamente o impacto da alocação vertical de recursos no desempenho de aplicações de alta demanda, como benchmarks científicos.

III. Explorando a Elasticidade em Contêineres

O controlador de elasticidade desenvolvido neste trabalho (Figura 1) foi projetado para operar de forma independente de orquestradores complexos, como o Kubernetes, visando cenários onde contêineres Docker são executados diretamente. Essa abordagem possibilita sua implantação em ambientes mais leves e com recursos limitados, como servidores de borda ou ambientes de teste, sem a sobrecarga e a complexidade de ferramentas de orquestração.

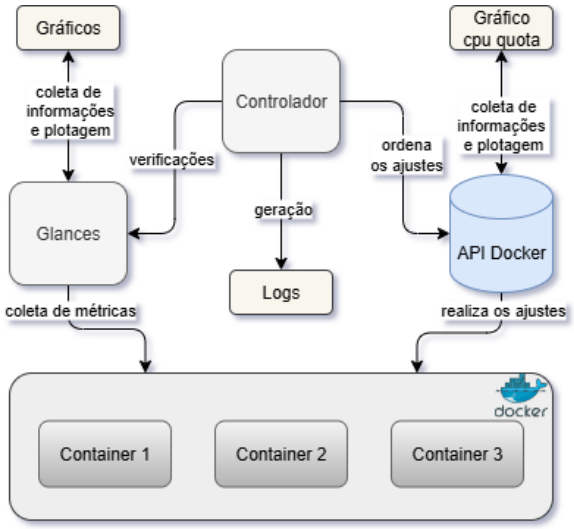

Fig. 1. Diagrama da arquitetura do controlador.

O comportamento do controlador pode ser resumido no pseudocódigo apresentado na Figura 2. O controle da elasticidade é realizado em duas etapas: (1) Coleta de métricas e (2) Tomada de decisão e ajuste dinâmico. Detalhes dos componentes são apresentados nas seções a seguir.

A. Coleta de métricas

A coleta de métricas é formada pelos módulos **Glances**, **Controlador** e **API Docker**. Esses módulos monitoraram, em tempo real, o uso de CPU e memória dos contêineres em execução. A coleta é realizada por meio da ferramenta Glances presente na máquina host. Como os dados fornecidos pela Glances não incluem a métrica de CPU quota, foi desenvolvido um script em Python que coleta especificamente essa informação de todos os contêineres em execução, utilizando a biblioteca *pandas* e a API do Docker. Essa abordagem garante a independência entre os métodos de coleta, permitindo capturar métricas em tempo real ao mesmo tempo em que mantém um histórico estruturado para análises posteriores.

```
Alocação Dinâmica

ler_parâmetros_configuração( )

Enquanto(true)
{
        coletar_info_contêineres (mem_info, CPU_info)
        ajustar_CPU( CPU_info )
        ajustar_memória( mem_info )
        registrar_log( )
}
```

Fig. 2. Algoritmo de alocação do controlador de elasticidade.

B. Tomada de decisão e ajuste dinâmico

As tarefas de tomada de decisão e ajuste dinâmico são realizadas pelo módulo **Controlador**, que aplica um conjunto de regras baseado em limiares pré-definidos, conforme Tabela I.

TABELA I
Conjunto de regras do controlador

Parâmetro	Valor
Intervalo de coleta	5 segundos
Intervalo de ajuste	5 segundos
Limites de CPU/RAM para subir	80%
Limites de CPU/RAM para descer	20%
Memória mínima absoluta	128 MB
CPU QUOTA alocada/desalocada	+100000 / -100000
RAM alocada/desalocada	+20% / -20%

As regras são avaliadas periodicamente a cada intervalo de coleta. A cada ciclo, o controlador verifica o consumo de CPU e memória de todos os contêineres em execução, detectando picos de utilização ou subutilização. Com base nesses dados, decide se é necessário alterar os recursos alocados. Os limiares são configuráveis, permitindo que o controlador se adapte a diferentes perfis de carga.

O ajuste de limites de CPU e memória é realizado dinamicamente por meio da **API do Docker**, sem interromper a execução dos contêineres. Para CPU, a quota é recalculada com base na utilização medida e no número de vCPUs atualmente alocadas, ajustando para mais ou para menos conforme os limites definidos. Para memória, o controlador aplica aumentos ou reduções percentuais, sempre respeitando um mínimo absoluto para evitar falhas por falta de recursos. Todas as alterações são registradas em logs, permitindo a análise posterior do comportamento do sistema.

IV. Resultados e Discussão

Resultados preliminares foram obtidos por meio da realização de dois experimentos: (1) teste de validação e (2) eficácia do controlador testada com uma aplicação real LULESH. O LULESH é um benchmark científico usado para simular a dinâmica de materiais sob choque, amplamente empregado em estudos de desempenho devido à sua alta demanda computacional e comportamento previsível [7]. O hardware da máquina utilizada para os testes possui 16 threads e 16 GB de memória RAM.

A. Teste de capacidade elástica

Para realizar esse experimento, foram criados 2 contêineres. O primeiro, **teste-incremento**, foi estressado utilizando o valor de 200000 de CPU quota, com o uso de 100% das duas vCPUs. A memória RAM do mesmo foi limitada a 2 GB, com a utilização de 1.7 GB, ou seja, 85% de utilização. Ele serve para testar a capacidade do controlador de aumentar os recursos do contêiner. O segundo, **teste-decremento**, é usado para testar a capacidade do mecanismo de reduzir os recursos alocados. Ele foi configurado com o valor de CPU quota de 400000, com carga de aproximada de 8% das 4 vCPUs. A memória RAM também foi limitada a 2 GB, mas com a utilização de 300 MB, ou seja, 15% do total.

Os resultados referentes à CPU são apresentados na Figura 3. No instante 13:33:45 em diante, que é quando o controlador é ativado, observa-se a alocação de CPU Quota em **teste-incremento** até o fim da execução. Em **teste-decremento**, observa-se o contrário: 3 vCPUs são desalocadas (de 4 para apenas 1), devido à subutilização.

Os resultados referentes à memória são apresentados na Figura 4. Do instante 13:33:45 em diante, o controlador age para aumentar o limite máximo de memória disponível para o contêiner **teste-incremento**, até que a porcentagem de uso seja menor que 80%. De modo inverso, no contêiner **teste-decremento** a memória é decrementada devido ao baixo uso, até chegar ao limite mínimo estabelecido (128 MB).

B. LULESH

Neste experimento, utiliza-se a aplicação LULESH multi-thread containerizada. O contêiner foi configurado com 2 GB de memória e 4 cores. Os parâmetros do tamanho do problema e do número de iterações foram, respectivamente, 25 e 800. A aplicação foi executada com e sem o uso do controlador, com foco no uso da CPU.

Sem o controlador ativo, obteve-se os seguintes resultados, apresentados na Figura 5. Observa-se no gráfico que o CPU Quota se manteve inalterado, e o uso da CPU varia pouco durante a execução, ficando muito próximo de 400% (uso de 4 vCPUs).

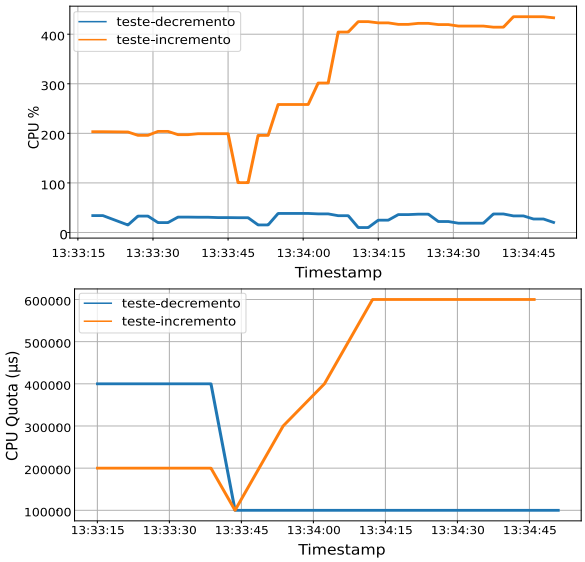

Fig. 3. Porcentagem de uso de CPU (cima). CPU quota (baixo)

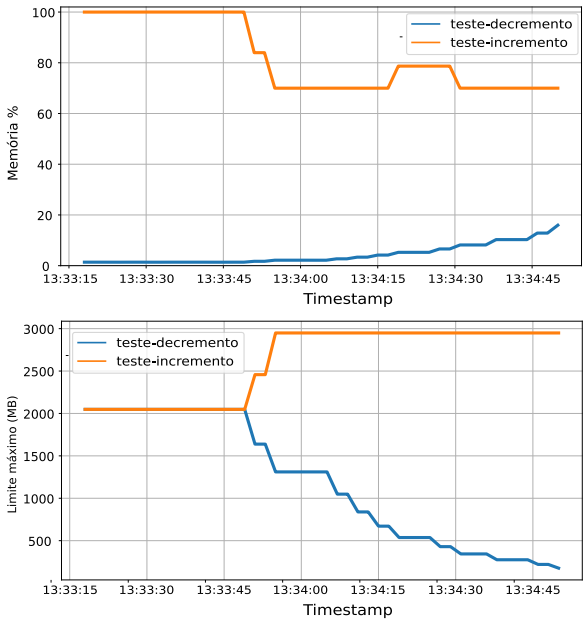

Fig. 4. Porcentagem de uso de memória (cima). Limite máximo de memória alocado (baixo)

Com o controlador ativo, os resultados são mostrados na Figura 6. Com o uso da elasticidade, percebe-se que a aplicação pode alocar mais recursos de CPU da máquina host, fazendo com que seja executada em um tempo cerca de 75% menor, como mostra a Figura 7.

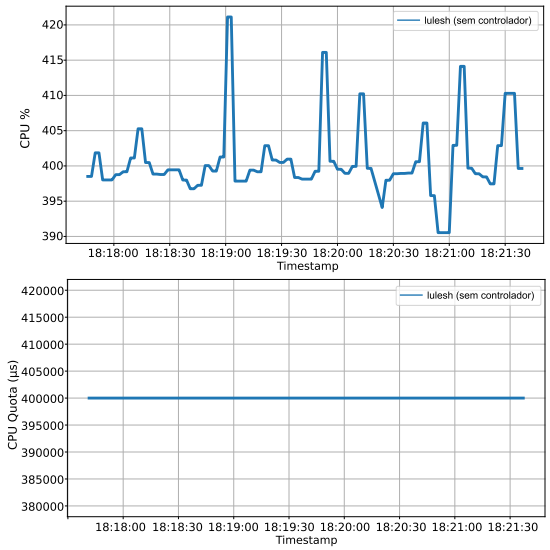

Fig. 5. Acima, LULESH sem elasticidade: Uso de CPU (%). Abaixo, CPU quota.

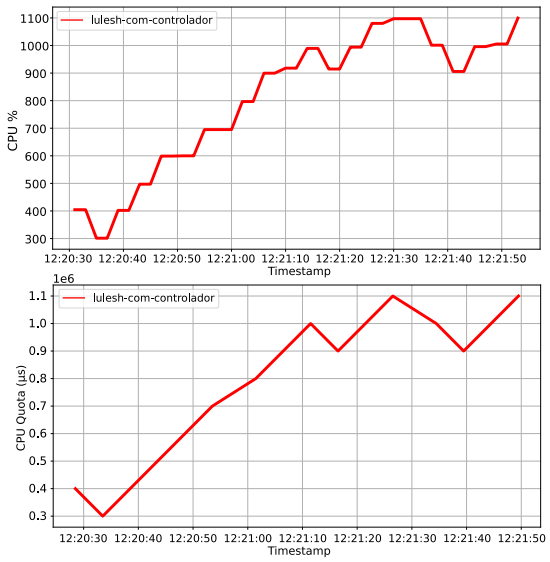

Fig. 6. Acima, LULESH com elasticidade: Uso de CPU (%). Abaixo, CPU quota.

V. Conclusão

Os resultados mostram que o controlador desenvolvido consegue monitorar e ajustar dinamicamente os recursos de CPU e memória de todos os contêineres em execução em uma máquina física. Pode-se observar que o controlador consegue manter o uso de CPU e memória dentro dos limites definidos, realizando ajustes precisos e contínuos; a abordagem adotada é viável e

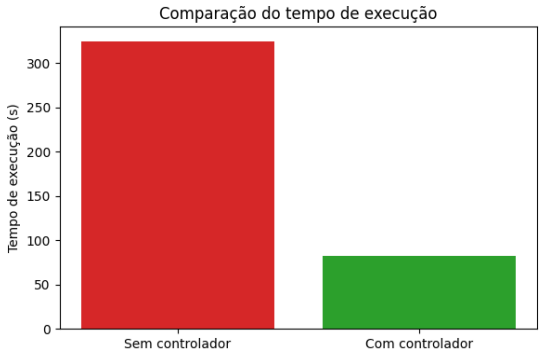

Fig. 7. Comparação do tempo de execução da aplicação.

cumpre a proposta definida, podendo melhorar o desempenho, escalabilidade e a confiabilidade das aplicações, bem como permitindo o uso eficiente da infraestrutura.

Como trabalhos futuros, pretende-se investigar a implementação de estratégias de elasticidade preditiva em contêineres, capazes de antecipar variações de carga e ajustar os recursos antes que ocorra a sobrecarga ou subutilização.

Agradecimentos

Agradecimentos à Universidade e ao Programa Institucional de Bolsas de Iniciação Científica (PIBIC) pelo suporte e recursos disponibilizados para a realização deste projeto.

Referências

[1] N. R. Herbst, S. Kounev, and R. Reussner, "Elasticity in cloud computing: What it is, and what it is not," in *10th International Conference on Autonomic Computing (ICAC 13)*. San Jose, CA: USENIX Association, Jun. 2013, pp. 23–27. [Online]. Available: https://www.usenix.org/conference/icac13/technical-sessions/presentation/herbst

[2] J. Dogani, R. Namvar, and F. Khunjush, "Auto-scaling techniques in container-based cloud and edge/fog computing: Taxonomy and survey," *Computer Communications*, vol. 209, pp. 120–150, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366423002086

[3] A. Barnawi, S. Sakr, W. Xiao, and A. Al-Barakati, "The views, measurements and challenges of elasticity in the cloud: A review," *Computer Communications*, vol. 154, pp. 111–117, 2020.

[4] M. A. N. Saif, S. K. Niranjan, and H. D. E. Al-ariki, "Efficient autonomic and elastic resource management techniques in cloud environment: Taxonomy and analysis," *Wireless Networks*, vol. 27, no. 4, pp. 2829–2866, 2021.

[5] J.-y. Choi, M. Cho, and J.-S. Kim, "Employing vertical elasticity for efficient big data processing in container-based cloud environments," *Applied Sciences*, vol. 11, no. 13, p. 6200, Jul. 2021. [Online]. Available: http://dx.doi.org/10.3390/app11136200

[6] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Autonomic vertical elasticity of docker containers with elasticdocker," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. IEEE, Jun. 2017, p. 472–479. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2017.67

[7] I. Karlin, J. Keasler, and R. Neely, "Lulesh 2.0 updates and changes," Tech. Rep. LLNL-TR-641973, August 2013.