# Uma Experiência de Execução de Aplicação MPI em Contêineres com Docker Swarm

**Bruno da Silva Alves, Andrea S. Charão, João Vicente F. Lima**

[1] Laboratório de Sistemas de Computação
Universidade Federal de Santa Maria

***Resumo.*** *A orquestração de contêineres é um dos principais desafios para solução de problemas que exigem alto desempenho. Nesse contexto, o presente trabalho propõe uma investigação visando adquirir respostas sobre o impacto da utilização do Docker Swarm na execução de aplicações MPI. Verificou-se que o Docker Swarm se apresenta como uma boa solução para orquestração de contêineres, porém ainda existem deficiências que devem ser tratadas.*

## 1. Introdução

Os contêineres do Linux caracterizam um método de virtualização em nível de sistema operacional e, atualmente, vêm se demonstrando como uma boa solução para os problemas que demandam alto desempenho, pois conseguem prover um bom nível de isolação com decréscimos leves no que diz respeito à performance. Pesquisadores em computação têm se dedicado a investigar o impacto de contêineres em aplicações de HPC. Cientistas de diversas áreas também já reconhecem o potencial dessa abordagem: por exemplo, a utilização de contêineres para a execução do WRF, uma aplicação de predição numérica de clima [Hacker et al. 2017].

Embora trabalhos apontem vantagens, há também desafios. Dentre os desafios está a orquestração, quando aplicações são paralelas e distribuídas como por exemplo aplicações MPI e OpenMP. O Docker Swarm surgiu como uma alternativa para a orquestração de contêineres, apresentando-se como uma solução de código aberto e integrada a Docker Engine.

Nesse contexto, o presente trabalho propõe uma investigação exploratória inicial, visando adquirir respostas sobre o impacto da utilização do Docker Swarm na execução de aplicações MPI. A fim de obter dados sobre tais impactos, são propostos quatro cenários para a execução do *NAS Parallel Benchmark*, uma aplicação MPI e OpenMP popular.

## 2. Trabalhos Relacionados

Vários artigos apresentaram estudos sobre a performance de contêineres. Antes do surgimento do Docker como uma plataforma comum para criação de contêineres, o trabalho de [Xavier et al. 2013] mostrou que os sistemas baseados em contêiner como LXC, OpenVZ e VServer demonstram uma performance de CPU, memória e acesso a disco similar ao sistema nativo XEN. Tal trabaho também fez o uso do *NAS Parallel Benchmarks* para os testes de performance. Após o surgimento do Docker, [Chung et al. 2016] chegou a conclusão que a utilização dos contêineres do Docker proporcionavam muitas vantagens como portabilidade e escalabilidade em relação ao uso do Máquinas Virtuais, e que eram mais adequados para aplicações com intensa troca de dados.

No que diz respeito a orquestração de contêineres, [Tosatto et al. 2015] realizou um estudo sobre os desafios na área de orquestração e concluiu que as pequenas empresas não provisionavam serviços com contêineres devido a falta de padronização e a dificuldade para se monitorar os mesmos. Porém, o Docker Swarm ainda não havia surgido como uma solução para orquestração. O artigo de [Nguyen e Bein 2017] apresenta uma solução para desenvolvimento e implementação de aplicações MPI em *clusters* de Docker contêineres orquestrados pelo Docker Swarm e demonstra que essa prática torna mais simples a solução de problemas de alto desempenho em clusters distribuidos. No entanto, o trabalho de [Nguyen e Bein 2017] não apresenta quais os impactos que o Docker Swarm pode causar na resolução de problemas dessa magnitude.

## 3. Docker Swarm

O modo Swarm do Docker consiste em um conjunto de ferramentas que permitem o gerenciamento e orquestração de servidores rodando a Engine do Docker. Dentre as principais vantagens na utilização do modo Swarm, pode-se destacar as seguintes: integração com a Docker Engine (permitindo o gerenciamento de Cluster sem softwares adicionais e com uma sintaxe de comandos similar), escalável (os estados dos serviços são mantidos quando o mesmos são escalados), possui balanceamento de carga e é um software de código aberto que conta com atualizações constantes.

Os nós que compõem o Swarm podem assumir três papéis: gerente, trabalhador ou gerente e trabalhor. Os nós configurados como gerente exercem as funções de delegação de tarefas e gerenciamento de associações, já os trabalhadores executam os serviços instanciados pelo Swarm. Ainda, o Swarm permite a criação de serviços, os quais definem um conjunto de tarefas a serem executadas nos nós participantes do Swarm. Antes do surgimento do Docker Swarm, algumas soluções para a orquestração de contêineres foram desenvolvidas como o Kubernets, CoreOS Fleet, Mesosphere Marathon [Nguyen e Bein 2017]. O presente trabalho faz uso do Swarm pelas vantagens descritas anteriormente.

## 4. Materiais e Métodos

Na composição do cenário de testes foram utilizadas três ferramentas principais: a Docker Engine, a biblioteca OpenMPI e o *NAS Parallel Benchmarks*. O *NAS Parallel Benchmarks* consiste em conjunto de programas desenvolvidos para avaliar a performance de supercomputadores paralelos. Dentre os benchmarks disponiveis no NAS encontra-se o FT, um programa que executa transformações discretas de Fourier. O programa FT foi escolhido devido a sua característica de realizar comunicações do tipo todos para todos, sendo assim exigente em termos de comunicações e ideal para testes envolvendo vários servidores. Ainda, os programas disponibilizados pelo NAS são categorizados em diferentes classes, as quais definem o tamanho do problema que será executado. Os testes foram realizados com problemas de tamanho padrão (A, B e C) e que aumentam quatro vezes de tamanho entre uma classe e outra (da classe A para a classe C). A versão do NAS utilizada foi a NPB-3.3.1.

A máquina utilizada para a coleta de dados é uma NUMA SGI UV2000 de arquitetura com 48 núcleos de processamento (Intel® Xeon® CPU E5-4617 @ 2.90GHz) e 512GB de memória RAM. As execuções paralelas realizadas usam MPI com a versão 2.1.5 da distribuição OpenMPI.

O benchmark FT foi executado em quatros cenários distintos: a, b, c e d. No cenário (a) foram configurados 16 contêineres rodando uma modificação da imagem do ubuntu:14.04 que continha chaves RSA para a validação do acesso SSH, necessário para a execução do comando *mpi-run*. Tal imagem ainda continha o benchmark FT compilado para a execução paralela em 16 nós do MPI. A criação dos 16 contêineres do cenário (a) se deu através do comando *docker-compose*. O cenário (b) faz o uso do Docker Swarm para a criação dos 16 contêineres que rodavam a mesma imagem do cenário (a). Assim, se fez necessário criar uma rede de sobreposição para conectar todos os contêineres, a rede foi criada com a ferramenta docker network. As redes de sobreposição são importantes na execução do Docker Swarm pois permitem a comunicação entre contêineres rodando em nós posicionados em redes distintas.

O cenário (c) é uma modificação do cenário (a), que consiste na execução dos contêineres com a *flag –cpuset-cpus* que permite determinar em quais *cores* o contêiner deve utilizar. Assim, cada contêiner foi configurado para ter acesso exclusivo a um *core* da máquina, esse tipo de configuração não está disponível com o uso do Swarm. O cenário (d) é a execução paralela do programa FT no sistema operacional da máquina Intel Xeon.

## 5. Resultados e Discussão

Em um primeiro momento, foi realizada a coleta de dados do tempo de execução a partir dos cenários (a) e (b), a fim de examinar o comportamento da aplicação em diferentes situações de configurações. Os valores obtidos são apresentados na Figura 1 e são referentes a uma média aritmética de 10 execuções do programa FT nas classes A, B e C. A partir destes dados, é possível verificar que os valores obtidos com o uso dos contêineres nas configurações padrão (cenário (a)) e os valores com o uso do Docker Swarm (cenário (b)) assemelham-se nos diferentes tipos de execuções testados.

No entanto, durante a execução dos testes com os cenários (a) e (b) notou-se, através do uso da ferramenta docker stats, que os contêineres não utilizavam 100% do uso de CPU, tanto com a utilização do Swarm quanto sem a utilização do mesmo, nas configurações padrão dos contêineres. Dessa forma, o cenário (c) foi proposto com a finalidade de avaliar se com o uso de alocação de recursos seria possível aumentar a performance na execução de programas. Na comparação entre os cenários (c) e (d) é possível perceber que o uso dessa configuração diminui a performance da aplicação em níveis aceitáveis em relação aos benefícios proporcionados pelo uso de contêineres, como portabilidade e escalabilidade.

## 6. Considerações Finais

A partir dos testes realizados neste trabalho, pôde-se observar um comportamento do Docker Swarm, que até então não havia sido investigado. Tais resultados comprovam que a utilização do Docker Swarm não gera impactos negativos significativos de performance em relação a execução somente com contêineres na configuração padrão, sem orquestração. O uso Docker Swarm proporciona uma solução para orquestração de contêineres no contexto de aplicações MPI de alto desempenho. Porém, através dos cenários testados foi possível verificar que o Docker Swarm ainda não permite a alocação de cores específicos para cada contêiner criado. A alocação dos cores permitiu que o desempenho da aplicação chegasse a níveis mais próximos ao desempenho nativo. Portanto, para o Docker Swarm

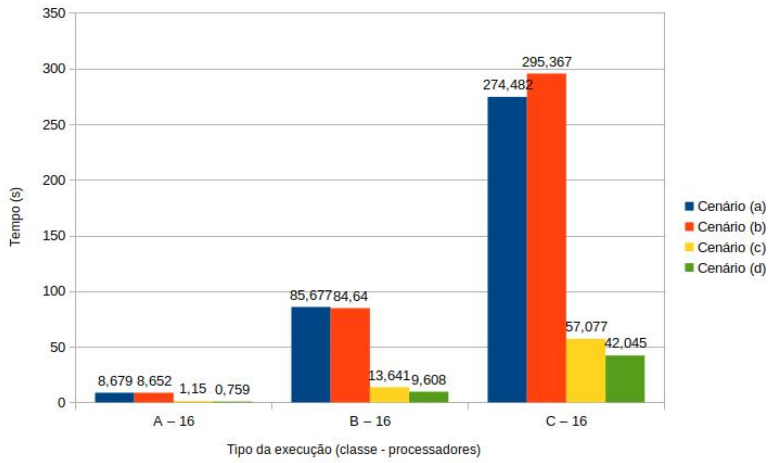


**Figura 1. Tempo de execução do *Benchmark* FT em diferentes cenários.**

se afirmar como uma solução padrão de orquestração é necessário que o mesmo inclua ferramentas de alocação de recursos. Como trabalho futuro pretende-se avaliar a performance dos contêineres na execução de outros *benchmarks* do pacote NAS, testar quais outras configurações que podem melhorar a performance da aplicação e, ainda, verificar se tais configurações estão presentes tanto para o uso dos contêiners do Docker quanto para o uso com o Docker Swarm.

## 7. Agradecimentos

Este trabalho foi parcialmente financiado pelo projeto "GREEN-CLOUD: Computação em Cloud com Computação Sustentável" (#162551-0000 488-9), no programa FAPERGS-CNPq PRONEX 12/2014.

## Referências

Chung, M. T., Quang-Hung, N., Nguyen, M., e Thoai, N. (2016). Using docker in high performance computing applications. In *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, pages 52–57.

Hacker, J. P., Exby, J., Gill, D., Jimenez, I., Maltzahn, C., See, T., Mullendore, G., e Fossell, K. (2017). A containerized mesoscale model and analysis toolkit to accelerate classroom learning, collaborative research, and uncertainty quantification. *Bulletin of the American Meteorological Society*, 98(6):1129–1138.

Nguyen, N. e Bein, D. (2017). Distributed mpi cluster with docker swarm mode. In *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1–7.

Tosatto, A., Ruiu, P., e Attanasio, A. (2015). Container-based orchestration in cloud: State of the art and challenges. In *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, pages 70–75.

Xavier, M. G., Neves, M. V., Rossi, F. D., Ferreto, T. C., Lange, T., e Rose, C. A. F. D. (2013). Performance evaluation of container-based virtualization for high performance computing environments. In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 233–240.