

Containerização do Modelo WRF em Arquiteturas Heterogêneas

Guilhermy F da Silva², João Vítor P Martins², Roussian R A Gaioso²,
Angel D Chovert, Ricardo A P Franco¹, Wellington S Martins¹

¹Universidade Federal de Goiás (UFG) – Goiânia, GO – Brasil

²Pontifícia Universidade Católica de Goiás (PUC-GO) – Goiânia, GO – Brasil

{joaovitorpei62, guilhermyfeleila}@gmail.com

roussian@pucgoias.br

{angel.chovert, ricardofranco, wsmartins}@ufg.br

Abstract. Containers are widely adopted in HPC for reproducibility, under the premise that an image built once runs anywhere. This work evaluates that premise for the WRF atmospheric model across four environments with distinct microarchitectures. We show that portability is bounded by two couplings the abstraction does not remove: the compile-time target microarchitecture and the MPI stack, whose launcher on the host must remain protocol-compatible with the library in the image. Each coupling manifests either as an explicit failure or as unused instruction-set capability, which motivates a per-target build strategy.

Resumo. contêineres são adotados em HPC por reprodutibilidade, sob a premissa de que uma imagem construída uma vez executa em qualquer lugar. Este trabalho avalia essa premissa para o modelo atmosférico WRF em quatro ambientes com microarquiteturas distintas. Mostra-se que a portabilidade é limitada por dois acoplamentos que a abstração não remove: a microarquitetura alvo, fixada em compilação, e a pilha de MPI, cujo lançador no host deve manter compatibilidade de protocolo com a biblioteca da imagem. Cada acoplamento se manifesta como falha explícita ou como capacidade de instruções não utilizada, o que motiva uma estratégia de construção por alvo.

1. Introdução

O *Weather Research and Forecasting* (WRF) [Skamarock et al. 2019] é um modelo atmosférico de referência em previsão numérica de tempo. Sua execução depende de uma cadeia extensa de bibliotecas, cuja compilação manual é demorada, sensível a versões e difícil de reproduzir entre máquinas distintas.

A containerização surge como resposta natural a esse problema. Trabalhos anteriores demonstraram que contêineres alcançam desempenho próximo ao nativo em cargas científicas [Xavier et al. 2013, Saha et al. 2018] e que a tecnologia é viável para o próprio WRF em nuvem [Powers et al. 2021, Duran-Limon et al. 2016]. Esses resultados consolidaram a expectativa de que, sendo desprezível o *overhead* de virtualização, uma imagem construída uma única vez poderia ser executada em qualquer infraestrutura. É essa

segunda parte, a portabilidade, que este trabalho problematiza. A literatura avalia predominantemente cenários homogêneos, ao passo que o caso estudado abrange desenvolvimento em servidor Intel de geração antiga, experimentação em aglomerado acadêmico com nós AMD e Intel distintos e produção em nuvem AMD sem *runtime* de contêiner disponível. Nesse ambiente, a premissa de imagem única não se sustenta.

2. Materiais e Métodos

A Tabela 1 descreve os quatro ambientes utilizados.

Tabela 1. Ambientes de execução

Ambiente	Processador	Microarq.	Runtime
Desenvolvimento	Intel Xeon E5-2697 v2	Ivy Bridge	Docker
Experimentação	AMD EPYC 7452	Zen2	Apptainer
Experimentação	Intel Xeon Gold 6148	Skylake-SP	Apptainer
Produção	AMD EPYC Milan	Zen3	nenhum

Utilizou-se o WRF 4.4.2 sobre o ciclo GFS de 24 de março de 2026, 00Z. O domínio cobre a região Sul-Sudeste do Brasil, em projeção Lambert Conformal, com dois domínios aninhados de 9 km (325×348) e 3 km (604×763), 36 níveis verticais e horizonte de previsão de 3 horas. As imagens foram construídas com Docker e convertidas para o formato Apptainer no aglomerado.

Em execuções multinó, o lançamento dos processos é feito pelo *runtime* de MPI do host, enquanto o binário e a biblioteca de MPI residem no contêiner.

3. Resultados

3.1. Acoplamento à implementação de MPI

A Tabela 2 apresenta os tempos obtidos sob descasamento de versão entre a biblioteca de MPI do contêiner e o módulo de lançamento do host.

Tabela 2. Tempos sob condição divergente (contêiner: Intel MPI 2021.10; host: 2021.13). A condição correspondente permanece em execução.

Configuração	Nós	Processos	Tempo (s)
MPI divergente	2	80	744 ± 3

O valor corresponde à média de cinco repetições da mesma configuração, mantidos constantes o caso de simulação, a contagem de processos e a alocação de nós. O modelo completou a simulação e gerou os arquivos de saída esperados em todas as execuções, incluindo as multinó, sem que o descasamento de versão menor produzisse erro em tempo de execução.

Registrou-se, em contraste, falha por instrução ilegal ao utilizar os módulos MPICH do sistema nos nós AMD, uma vez que tais módulos foram compilados para hardware Intel. Nesse caso, o acoplamento se manifesta como erro explícito, e não como perda silenciosa.

A quantificação do custo do descasamento exige a condição correspondente, isto é, a mesma imagem sob módulo de host de versão idêntica, e permanece como trabalho em andamento.

3.2. Acoplamento à microarquitetura

A opção `-march=native` converte a máquina de construção em dependência oculta da imagem. Um binário gerado no ambiente de desenvolvimento não emprega AVX2 nos nós Zen2; um binário com AVX-512 gerado no Skylake-SP falha por instrução ilegal em hardware AMD. A consequência é que a imagem deixa de ser única: cada alvo exige compilação própria (`znver2`, `znver3`, `skylake-avx512`, `core-avx2`).

Obstáculos adicionais decorrem dessa exigência. O GCC 9.4, padrão da distribuição-base, não reconhece `-march=znver3`, demandando a adoção do GCC 11, e o ambiente de produção exigiu vinculação estática das bibliotecas de *runtime*.

4. Conclusão

contêineres entregam reprodutibilidade de ambiente, não portabilidade binária. Em HPC, a imagem permanece acoplada à microarquitetura alvo e à pilha de MPI do hospedeiro. Ignorar esse acoplamento leva, no melhor caso, ao desperdício das extensões vetoriais do hardware de execução e, no pior, à impossibilidade de execução por instrução ilegal. A estratégia de construção por alvo preserva a reprodutibilidade ao custo de múltiplas imagens. Como trabalho em andamento, prossegue-se com a quantificação do ganho de cada alvo de microarquitetura e com a medição do custo do descasamento de MPI sob as condições correspondente e divergente.

Agradecimentos

Este trabalho contou com o apoio do Laboratório Multiusuário de Computação de Alto Desempenho (LaMCAD) da UFG, do Centro de Excelência em Inteligência Artificial (CEIA) da UFG, e da startup i4sea.

Referências

- Duran-Limon, H. A., Flores-Contreras, J., Parlavantzas, N., Zhao, M., and Meulenert-Peña, A. (2016). Efficient execution of the WRF model and other HPC applications in the cloud. *Earth Science Informatics*, 9(3):365–382.
- Powers, J. G. et al. (2021). Cloud computing efforts for the weather research and forecasting model. *Bulletin of the American Meteorological Society*, 102(6).
- Saha, P., Beltre, A., Uminski, P., and Govindaraju, M. (2018). Evaluation of Docker containers for scientific workloads in the cloud. In *Proceedings of the Practice and Experience on Advanced Research Computing (PEARC '18)*. ACM.
- Skamarock, W. C., Klemp, J. B., Dudhia, J., et al. (2019). A description of the advanced research WRF model version 4. Technical report, National Center for Atmospheric Research.
- Xavier, M. G., Neves, M. V., Rossi, F. D., Ferreto, T. C., Lange, T., and De Rose, C. A. F. (2013). Performance evaluation of container-based virtualization for high performance computing environments. In *21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 233–240. IEEE.