

# Introdução à Reprodutibilidade de Experimentos Computacionais de Alto Desempenho

Vinícius Garcia Pinto, Lucas Leandro Nesi, Lucas Mello Schnorr

Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS)  
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{vgpinto, llnesi, schnorr}@inf.ufrgs.br

***Resumo.** Este tutorial introduz uma metodologia reprodutível para realização de experimentos na área de processamento paralelo de alto desempenho. O objetivo é motivar a adoção de boas práticas experimentais que possibilitem a coleta de medidas mais representativas e que por consequência levem a resultados mais confiáveis. A metodologia apresentada é organizada em duas fases: execução de experimentos e análise dos dados. São sugeridas técnicas e ferramentas apropriadas para cada uma das etapas que compõe estas duas fases.*

## 1. Introdução

Reprodutibilidade é um aspecto chave da pesquisa científica que possibilita que observações e experimentos possam ser refeitos de maneira independente. No área de processamento de alto desempenho, é natural lidar com múltiplos computadores que por vezes também são especializados, ou seja, contêm placas aceleradoras e interfaces de rede com grande largura de banda e baixa latência.

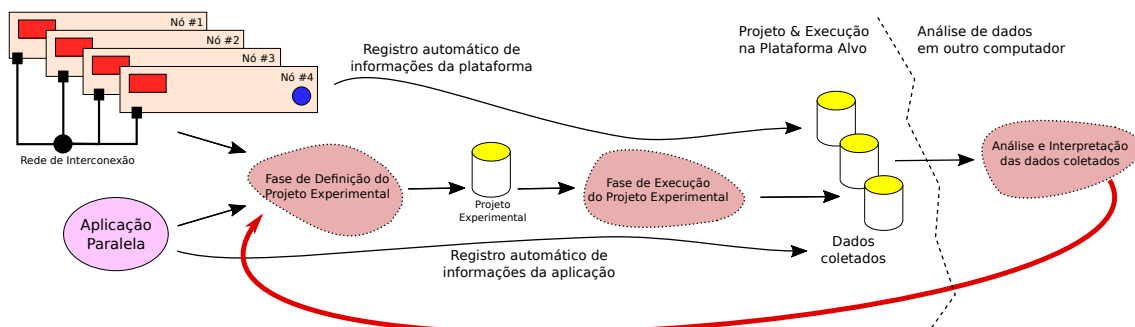
Experimentos computacionais confiáveis precisam envolver a coleta de medidas representativas. Isso implica registrar e, quando possível, controlar a configuração de *hardware* e *software* usada nos testes. Idealmente cada resultado produzido deve ser acompanhado do respectivo registro da plataforma, permitindo identificar e correlacionar aqueles aspectos que possam influenciar o experimento em questão, tais como atualizações de *software* ou melhorias no *hardware*. Manter esse fluxo de trabalho torna os experimentos naturalmente mais rigorosos, exigindo maior atenção. Este esforço pode ser diluído por meio da criação de *scripts* executados automaticamente no momento da execução de cada experimento. Tal disciplina estende-se também ao posterior trabalho nos dados coletados como obtenção de medidas de desempenho, tais como médias, mínimos, máximos, dispersão e na elaboração de gráficos e tabelas.

Este minicurso aborda uma introdução à reprodutibilidade de experimentos computacionais de alto desempenho. Apresentamos as técnicas e conceitos com o objetivo de motivar a adoção de procedimentos disciplinados, que levem a conclusões que não apenas evidenciem os efeitos observados mas que sejam portáteis no tempo (futuras observações) e no espaço (outras plataformas computacionais semelhantes). Uma abordagem aprofundada é discutida em [Pinto et al. 2020]; materiais e referências complementares são catalogados em: <https://exp-hpc.gitlab.io/>.

## 2. Visão Geral da Metodologia

A metodologia para experimentos reprodutíveis se inspira no trabalho de Jain [Jain 1991]. A Figura 1 ilustra a sua organização em duas fases. A partir dos objetivos estabelecidos

para investigação, define-se o projeto experimental com **fatores** (de controle) e **variáveis de resposta**. A **execução dos projeto experimental** (esquerda da linha pontilhada) envolve também a coleta automatizada de informações da plataforma e da aplicação por meio de *scripts*. O projeto combina, replica e torna aleatória a ordem de execução para reduzir o impacto do indeterminismo de anomalias durante os experimentos. Ao final dos experimentos, os dados coletados compreendem tanto informações constantes da aplicação e da plataforma registrados, assim como dados de entrada e de saída da execução.



**Figura 1. Metodologia experimental com enfoque na reprodutibilidade.**

A **análise dos dados** (direita da linha pontilhada na 1) coletados é realizada após a execução da bateria experimental. É um processo que se inicia com uma análise geral dos resultados observados ao qual se segue um aprofundamento em pontos específicos. É comum que tais aprofundamentos revelem novos pontos a serem investigados e que, por consequência, novas baterias experimentais surjam a partir de observações de experimentos anteriores, fazendo da análise um **processo cíclico**. Cabe ressaltar que a automatização por *scripts* não desaparece totalmente, devendo ser aplicada nas etapas de transformação dos dados brutos e na geração de gráficos. Idealmente, tal código deve acompanhar as reflexões permitindo que se possa revisitar o fluxo de ideias que levou a uma dada conclusão.

### 3. Implementação da Metodologia

A metodologia descrita requer a coleta de informações da plataforma e da aplicação, o gerenciamento da pilha de software e o controle da plataforma. Várias ferramentas e configurações já existem para realizar todos estes processos. Nós propomos listas (não exaustivas) para elas. Todas as ferramentas estão catalogadas e referenciadas no nosso *companion*<sup>1</sup>. Para a coleta de informações da plataforma e da aplicação sugerimos as seguintes ferramentas: **lstopo** (do pacote *hwloc*) que mostra a topologia de hardware do sistema. **cpufreq-info** (*cpufrequtils*) que mostra informações como frequência dos processadores. **nvidia-smi** um comando equivalente para aceleradores NVIDIA. **lspci** (*Unix*) que apresenta informações dos dispositivos PCI. **ip**, **ifconfig**, ou outra ferramenta dependendo do gerenciador de rede, para informações sobre as interconexões de rede.

Devido a complexidade da pilha de software das aplicações, sugerimos ferramentas para a instalação e controle de dependências. Um exemplo é a ferramenta **spack** [Gamblin et al. 2015]. Para obter informações da aplicação os seguintes comandos podem ser úteis. **ldd** para listar as bibliotecas compartilhadas utilizadas. **env** para mostrar

<sup>1</sup><https://gitlab.com/exp-hpc/boas-praticas>

as variáveis de ambiente do sistema. **nm** para listar os símbolos da aplicação. **ompi-info** para mostrar as informações do *middleware* OpenMPI.

A execução dos experimentos é vastamente influenciada pela plataforma computacional utilizada e suas configurações. Para diminuir e controlar a variabilidade causada pela plataforma sugerimos: (i) **Controle da Frequência** dos processadores e aceleradores. (ii) **Vinculação (*Binding*)** dos fluxos de execução do programa (*threads*) nas unidades de processamento. (iii) **Desativar recursos de hardware** que podem causar variabilidade como *Intel TurboBoost*, *AMD Turbo Core* e *Simultaneous Multithreading (Intel HyperThreading)*. (iv) **Considerar o fator NUMA**, configurando as *threads* e dados conforme a topologia. (v) **Configurar a interface de interconexão**, como por exemplo as configurações padrões de TCP/IP. (vi) **Configurações do Kernel**, como *Address space layout randomization (ASLR)*, escalonadores e prioridades dos processos, e *driver* dos recursos.

A metodologia sugere a utilização da filosofia *literate programming* [Knuth 1984] que faz o uso de *journals* e *notebooks*. Estes documentos contêm linguagem natural descrevendo as atividades, blocos de código para a programação, além de dados e imagens para as análises de dados. Ambientes como **emacs + ess + org-mode**, **RStudio**, e **Jupyter Notebook** são algumas opções. A etapa de *data science* pode fazer o uso de ferramentas consagradas como a linguagem de programação **R**, com a biblioteca **tidyverse**, a linguagem de programação **Python**, com a biblioteca **Pandas**, ou a linguagem de programação **Julia**. Para a criação de *design de experimentos* os seguintes pacotes/bibliotecas podem ser utilizados: **DoE.base** (R), **FrF2** (R), **pyDOE2** (Python) e **ExperimentalDesign** (Julia).

## 4. Conclusão

Este texto apresenta uma breve introdução à reprodutibilidade de experimentos computacionais com objetivo de motivar o emprego de boas práticas pelo público de processamento paralelo de alto desempenho. A metodologia discutida abrange tanto a etapa de projeto e coleta de resultados experimentais quanto a posterior análise dos mesmos. Para cada etapa destes processos, são listados, a título de exemplo, algumas sugestões de procedimentos e ferramentas que podem ser usados para implementação da metodologia.

Uma vez que experimentos diferentes possuem requisitos e particularidades distintas, alguns dos pontos aqui discutidos podem ser insuficientes ou inaplicáveis. Espera-se entretanto que o presente material sirva de motivação e como ponto de partida para adoção de práticas que levem a resultados mais confiáveis.

## Agradecimentos

Este trabalho foi realizado com o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - Finance Code 001, do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) com a bolsa para o 2º autor (141971/2020-7), e dos projetos: FAPERGS ReDaS (19/711-6), MultiGPU (16/354-8) e GreenCloud (16/488-9), do projeto CNPq 447311/2014-0, do projeto CAPES/Brafitec 182/15 e CAPES/Cofecub 899/18, e com apoio do projeto Petrobras (2018/00263-5).

## Referências

- Gamblin, T., LeGendre, M., Collette, M. R., Lee, G. L., Moody, A., de Supinski, B. R., and Futral, S. (2015). The spack package manager: Bringing order to hpc software chaos. In *High Performance Computing, Networking, Storage and Analysis, 2015 SC-International Conference for*, pages 1–12. IEEE.
- Jain, R. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley.
- Knuth, D. E. (1984). Literate Programming. *The Computer Journal*, 27(2):97–111.
- Pinto, V., Nesi, L., and Schnorr, L. (2020). Boas Práticas para Experimentos Computacionais de Alto Desempenho. In du Bois, A. and Castro, M., editors, *Minicursos da XX Escola Regional de Alto Desempenho da Região Sul*, chapter 1, pages 1–19. SBC, Porto Alegre.