

# Introdução à Reprodutibilidade de Experimentos Computacionais de Alto Desempenho

## Tutorial 1

---

Vinícius Garcia Pinto

Lucas Leandro Nesi

Lucas Mello Schnorr

19 de agosto de 2020



# **Apresentação**

---

## 1. Baixar e lançar o container

---

```
docker pull viniciusvgp/boas-praticas:erad20  
docker run -it viniciusvgp/boas-praticas:erad20
```

---

## 1. Copiar de/para o container

---

```
docker ps # permite recuperar o containerID  
docker cp foo.txt <containerID>:/home/user  
docker cp <containerID>:/home/user/foo.txt .
```

---

Grande maioria dos trabalhos envolve experimentos computacionais

- Utilizam pelo menos uma máquina
- Podem envolver vários computadores

Grande maioria dos trabalhos envolve experimentos computacionais

- Utilizam pelo menos uma máquina
- Podem envolver vários computadores

Necessidade de analisar os dados, as medições, é onipresente

- Por vezes, a quantidade de dados é pequena
- Podem envolver muitos gigabytes de dados

Grande maioria dos trabalhos envolve experimentos computacionais

- Utilizam pelo menos uma máquina
- Podem envolver vários computadores

Necessidade de analisar os dados, as medições, é onipresente

- Por vezes, a quantidade de dados é pequena
- Podem envolver muitos gigabytes de dados

Análise de desempenho por vezes é feita de maneira superficial

- As conclusões não são necessariamente portáteis no tempo

# Objetivos

Sensibilizar participantes aos fatores que afetam a coleta de medidas

- Tornar os experimentos mais confiáveis
- Permitir conclusões mais perenes, mais ricas em informação

Sensibilizar participantes aos fatores que afetam a coleta de medidas

- Tornar os experimentos mais confiáveis
- Permitir conclusões mais perenes, mais ricas em informação

Detalhamento

1. Motivar cuidados essenciais em medidas computacionais
  - Controle da variabilidade experimental
2. Apresentar formas de controlar parâmetros em sistemas Linux
3. Como analisar dados de maneira reprodutível
4. Prática (em formato de tutorial auto-guiado com auxílio)

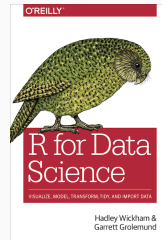
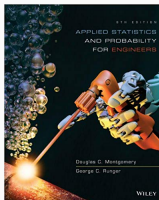
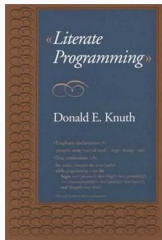


# Referência

Literate Programming. Donald E. Knuth.

Applied Statistics and Probability for Engineers. Montgomery & Runger.

R for Data Science. Golemund & Wickham.

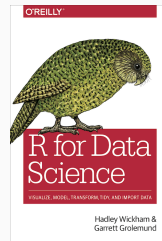
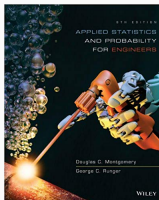
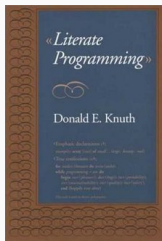


# Referência

Literate Programming. Donald E. Knuth.

Applied Statistics and Probability for Engineers. Montgomery & Runger.

R for Data Science. Golemund & Wickham.



Exemplos oriundos dos trabalhos do grupo de pesquisa

# Propaganda

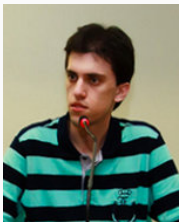
---

# Quem somos nós?

## Vinícius Garcia Pinto

Pós-doutorando e Prof.  
substituto - INF/UFRGS

<http://www.inf.ufrgs.br/~vgpinto>  
vgpinto@inf.ufrgs.br



## Lucas Leandro Nesi

Doutorando do PPGC -  
INF/UFRGS

<http://www.inf.ufrgs.br/~llnesi>  
llnesi@inf.ufrgs.br



## Lucas Mello Schnorr

Prof. INF/UFRGS &  
orientador PPGC

<http://www.inf.ufrgs.br/~schnorr>  
schnorr@inf.ufrgs.br



# Programa de Pós-Graduação em Computação (PPGC)

Site do programa: <<http://www.inf.ufrgs.br/ppgc/>>

Oferece

- Mestrado, entradas anuais em março
  - Edital é lançado no segundo semestre
  - Requisito fundamental é o Poscomp
- Doutorado, entrada com fluxo contínuo

Fatos marcantes

- Conceito máximo (7) pela CAPES
- Internacionalização da formação e da investigação
- Número de Doutores formados: 361
- Número de Mestres formados: 1647

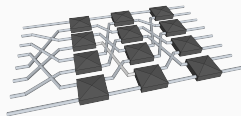


Site do grupo de pesquisa:

<<http://www.inf.ufrgs.br/gppd/site/>>

Eixos principais de investigação

- **High Performance Computing** (Computação de Alto Desempenho)
- Computer Architecture
- Big Data
- Cloud Computing
- FoG & Edge Computing

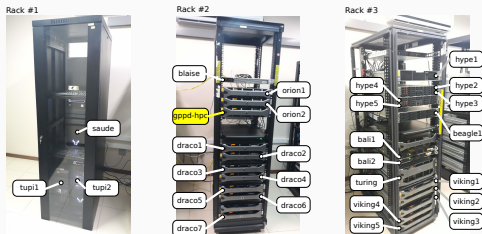


# Parque Computacional de Alto Desempenho (PCAD)

Site: <<http://gppd-hpc.inf.ufrgs.br/>>

Possui aproximadamente 30 nós: 700+ núcleos de CPU e 73.000+ de GPU

- Computação de alto desempenho heterogênea



Temos um GT (Grupo de Trabalho)

- Formamos alunos no gerenciamento destas plataformas

# Introdução

---



## Experimentos

validar ou refutar hipóteses

Confiabilidade → experimentos reprodutíveis

1. Exercer um controle sobre as variáveis controláveis
2. Registrar o valor das variáveis não controladas (contexto)

Experimentos em Sistemas Computacionais não são diferentes

- Único computador: software e hardware
- Cluster: todos os nós e a rede de interconexão

## Controle experimental

### **Desvantagens**

- Experimentos se tornam mais burocráticos
- Cuidado maior no antes, durante e depois dos experimentos
- Disciplina reforçada

## Controle experimental

### Desvantagens

- Experimentos se tornam mais burocráticos
- Cuidado maior no antes, durante e depois dos experimentos
- Disciplina reforçada
- Processo investigativo pode ser tornar mais lento

## Controle experimental

### Desvantagens

- Experimentos se tornam mais burocráticos
- Cuidado maior no antes, durante e depois dos experimentos
- Disciplina reforçada
- Processo investigativo pode ser tornar mais lento

### Vantagens

- Conclusões delineadas sejam mais perenes, significativas
- Relato facilitado (pois há substrato para derivar conclusões)
- Facilita a reprodutibilidade

## Automatização de tarefas

- Coleta dos dados por *scripts*
- Transformação/derivação de dados
- Preparação de estatísticas, gráficos e tabelas

# Boas práticas para experimentos em clusters HPC

## Automatização de tarefas

- Coleta dos dados por *scripts*
- Transformação/derivação de dados
- Preparação de estatísticas, gráficos e tabelas

## Características

- Impõem um cuidado na preparação da automatização
- Permite auditar o processo investigativo
- Trata-se de uma atividade multidisciplinar

Sistemas operacionais

Programação

Redes

Processamento Paralelo

Análise de dados

- Deve-se começar mesmo com uma estratégia simples

## 1. Teórica: Controle e Coleta

- Lista não exaustiva de controle de sistemas computacionais
- Projeto experimental

## 2. Teórica: Análise de Dados

- Como analisar os dados com ferramentas modernas de *data science*?
- Programação literária

## 1. Teórica: Controle e Coleta

- Lista não exaustiva de controle de sistemas computacionais
- Projeto experimental

## 2. Teórica: Análise de Dados

- Como analisar os dados com ferramentas modernas de *data science*?
- Programação literária

Prática, com quatro tutoriais curtos

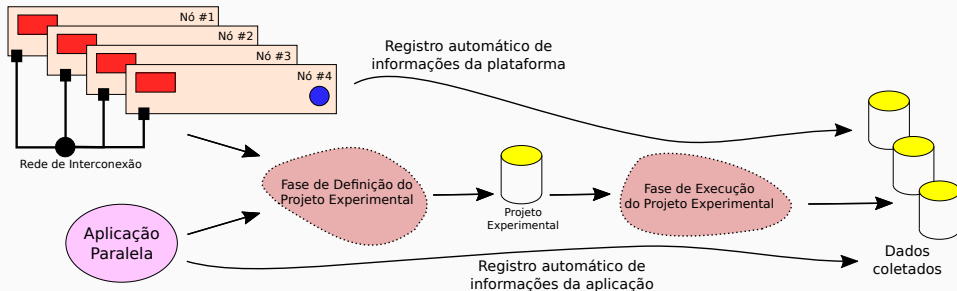
1. Instalação de ferramentas que permitem a rastreabilidade (`spack`)
2. Realização de experimentos computacionais (`bash`)
3. Análise de dados (`R+tidyverse`)
4. Criação de gráficos (`ggplot2`)



# Metodologia experimental em duas fases

## Fase 1 (Controle e Coleta)

Mecanismos automáticos guiados por um projeto experimental



## Fase 2 (Análise)

Mecanismos automáticos de tratamento dos dados

- Interpretação dos dados é feita *à posteriori*

Isolamento: força o experimentador a coletar bastante dados

## Controle e Coleta

---

# Características impactantes → aumento da variabilidade

## Fatores

- **Indeterminismo** da execução paralela (pela concorrência)
- Aparição de **anomalias** durante a execução
- **Complexidade** do sistema computacional

# Características impactantes → aumento da variabilidade

## Fatores

- **Indeterminismo** da execução paralela (pela concorrência)
- Aparição de **anomalias** durante a execução
- **Complexidade** do sistema computacional

## Aumento da Variabilidade dos experimentos

- Medidas são incertas, tem uma dispersão natural
- Quanto maior a dispersão, mais incertas são as conclusões

# Características impactantes → aumento da variabilidade

## Fatores

- **Indeterminismo** da execução paralela (pela concorrência)
- Aparição de **anomalias** durante a execução
- **Complexidade** do sistema computacional

## Aumento da Variabilidade dos experimentos

- Medidas são incertas, tem uma dispersão natural
- Quanto maior a dispersão, mais incertas são as conclusões
  - Exemplo: Medir o tempo de execução de uma aplicação paralela
    - Fazer várias execuções, calcular a **média**, calcular a **dispersão**

# Características impactantes → aumento da variabilidade

## Fatores

- **Indeterminismo** da execução paralela (pela concorrência)
- Aparição de **anomalias** durante a execução
- **Complexidade** do sistema computacional

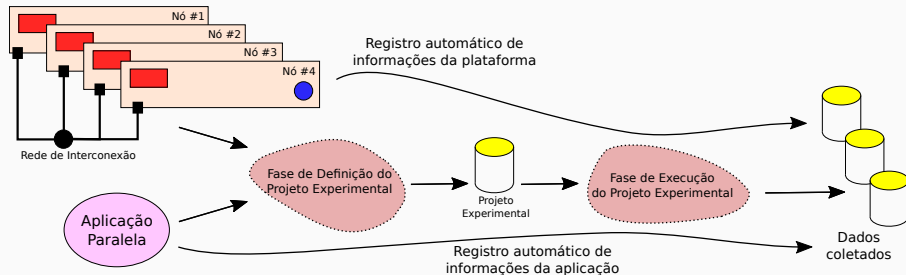
## Aumento da Variabilidade dos experimentos

- Medidas são incertas, tem uma dispersão natural
- Quanto maior a dispersão, mais incertas são as conclusões
  - Exemplo: Medir o tempo de execução de uma aplicação paralela
    - Fazer várias execuções, calcular a **média**, calcular a **dispersão**

Nada se pode fazer em relação ao indeterminismo e às anomalias

Resta tentar reduzir a complexidade do sistema computacional

# Controle e Coleta: Visão Geral



1. Metodologia experimental
2. Controle da complexidade
3. Registro de informações
4. Instalação de dependências
5. Controle em nível de sistema operacional



# 1. Metodologia experimental

## Projeto experimental

- Fatores (variáveis de controle) e níveis (seus valores)
- Variáveis de resposta (observações medidas)

# 1. Metodologia experimental

## Projeto experimental

- Fatores (variáveis de controle) e níveis (seus valores)
- Variáveis de resposta (observações medidas)

## Exemplo (aplicação paralela)

- 3× Variáveis de resposta: *makespan*, uso de energia, balanceamento
- 4× Fatores: qtdade processos, qtdade nós, freq. processador, rede

# 1. Metodologia experimental

## Projeto experimental

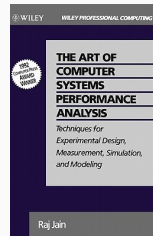
- Fatores (variáveis de controle) e níveis (seus valores)
- Variáveis de resposta (observações medidas)

## Exemplo (aplicação paralela)

- 3× Variáveis de resposta: *makespan*, uso de energia, balanceamento
- 4× Fatores: qtdade processos, qtdade nós, freq. processador, rede

## Leitura praticamente obrigatória

The Art of Computer Systems Performance Analysis:  
Techniques for Experimental Design, Measurement,  
Simulation, and Modeling by Raj Jain. Wiley, 1991.



# Tipos de projetos experimentais

Projetos simples, variam um único fator a cada vez

- Não permite o estudo da interação entre fatores

# Tipos de projetos experimentais

Projetos simples, variam um único fator a cada vez

- Não permite o estudo da interação entre fatores

**Projeto fatorial completo**, todas as combinações possíveis de fatores

- Permite estudo das interações
- Mais caro de ser executado (natureza combinatória)

# Tipos de projetos experimentais

Projetos simples, variam um único fator a cada vez

- Não permite o estudo da interação entre fatores

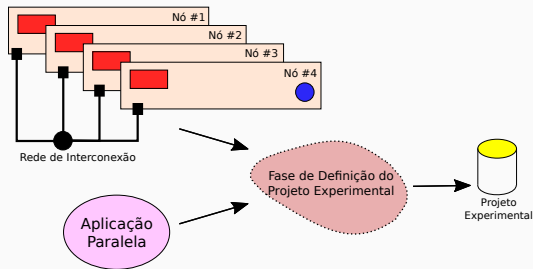
**Projeto fatorial completo**, todas as combinações possíveis de fatores

- Permite estudo das interações
- Mais caro de ser executado (natureza combinatória)

**Fatorial fracionário**

- Uma alternativa mais simples (sempre com um *trade-off*)

# Definir um projeto experimental



## Usando a linguagem R

```
library(DoE.base);  
exp0 <- fac.design (  
  nfactors=4,  
  replications=10,  
  randomize=TRUE,  
  factor.names=list(  
    process = c  
      (1,2,4,8,16,32,64,128,256),  
    node = c(1,2,4,8),  
    freq = c(1.2,1.8,2.1),  
    net = c("1GB","10GB","40GB"))
```

# Projeto experimental definido

O projeto é registrado em um arquivo CSV

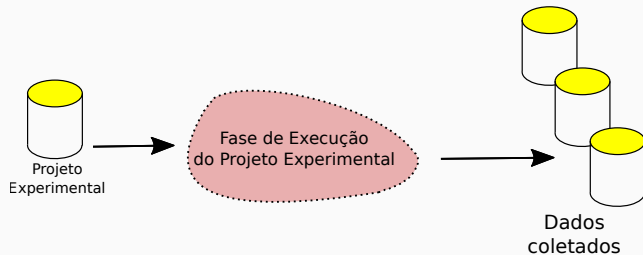
- Cada linha representa uma determinada configuração
- Valores das células representam os níveis dos fatores
- Ordem aleatória absorve o impacto das anomalias

```
head(exp0, n=9)
```

	process	no	freq	net	Blocks
1	64	4	1.2	10GB	.1
2	4	4	2.1	10GB	.1
3	128	4	1.2	1GB	.1
4	2	1	1.2	1GB	.1
5	256	1	1.2	1GB	.1
6	128	4	1.2	40GB	.1
7	16	8	2.1	10GB	.1
8	1	1	2.1	1GB	.1
9	4	4	1.2	1GB	.1



# Execução do projeto experimental



Programa de computador (*script* em *bash* por exemplo)

1. Ler o projeto experimental (no arquivo CSV)
2. Para cada linha do projeto, **executa a aplicação**
  - 2.1 Parâmetros tem origem nos fatores
  - 2.2 Coleta e registra as variáveis de resposta
3. Organiza os resultados em um diretório específico

## 2. Controle da complexidade

Diminuir a variabilidade → quais configurações fixar? Depende.

Listagem não exaustiva

- Vinculação fixa de fluxos de execução (*binding*)
  - Evita migração automática por algoritmos do SO
  - Difícil rastrear o comportamento da migração
  - Detectar e considerar a configuração NUMA

## 2. Controle da complexidade

Diminuir a variabilidade → quais configurações fixar? Depende.

Listagem não exaustiva

- Vinculação fixa de fluxos de execução (*binding*)
  - Evita migração automática por algoritmos do SO
  - Difícil rastrear o comportamento da migração
  - Detectar e considerar a configuração NUMA
- Controle da frequência dos núcleos de processamento
  - Fixar na frequência máxima (*governor userspace*)
  - Desabilitar *turboboost* (Intel), *turbo core* (AMD)
    - Não há como saber quando é ativado ou desativado

## 2. Controle da complexidade

Diminuir a variabilidade → quais configurações fixar? Depende.

Listagem não exaustiva

- Vinculação fixa de fluxos de execução (*binding*)
  - Evita migração automática por algoritmos do SO
  - Difícil rastrear o comportamento da migração
  - Detectar e considerar a configuração NUMA
- Controle da frequência dos núcleos de processamento
  - Fixar na frequência máxima (*governor userspace*)
  - Desabilitar *turboboost* (Intel), *turbo core* (AMD)
    - Não há como saber quando é ativado ou desativado
- Desativar SMT, *hyperthreading* (Intel)
  - Evitar uso de *cores* com recursos mais limitados
  - Especialmente importante em aplicações limitadas pela CPU

## 2. Controle da complexidade

Diminuir a variabilidade → quais configurações fixar? Depende.

Listagem não exaustiva

- Vinculação fixa de fluxos de execução (*binding*)
  - Evita migração automática por algoritmos do SO
  - Difícil rastrear o comportamento da migração
  - Detectar e considerar a configuração NUMA
- Controle da frequência dos núcleos de processamento
  - Fixar na frequência máxima (*governor userspace*)
  - Desabilitar *turboboost* (Intel), *turbo core* (AMD)
    - Não há como saber quando é ativado ou desativado
- Desativar SMT, *hyperthreading* (Intel)
  - Evitar uso de *cores* com recursos mais limitados
  - Especialmente importante em aplicações limitadas pela CPU
- Configurar uma política TCP/Ethernet adequada para a rede
  - Parâmetros *default* no Linux são para redes 100MBit

### 3. Registro automático de informações

Deve-se automatizar o registro de informações do sistema

- Coletadas toda vez que um experimento for realizado
- Armazenadas juntamente com os resultados do experimento

### 3. Registro automático de informações

Deve-se automatizar o registro de informações do sistema

- Coletadas toda vez que um experimento for realizado
- Armazenadas juntamente com os resultados do experimento

#### HW (Plataforma)

- ☐ `lstopo`, da ferramenta `hwloc`  
(topologia do sistema)
- ☐ `cpufreq-info` (frequência atual,  
mínima, máxima, governor)
- ☐ `ip` (ou `ifconfig`, obter configurações  
da interface de rede)
- ☐ `lspci` (todos os dispositivos PCI)

### 3. Registro automático de informações

Deve-se automatizar o registro de informações do sistema

- Coletadas toda vez que um experimento for realizado
- Armazenadas juntamente com os resultados do experimento

#### HW (Plataforma)

- ☐ `lstopo`, da ferramenta `hwloc` (topologia do sistema)
- ☐ `cpufreq-info` (frequência atual, mínima, máxima, governor)
- ☐ `ip` (ou `ifconfig`, obter configurações da interface de rede)
- ☐ `lspci` (todos os dispositivos PCI)

#### SW (Aplicação paralela)

- ☐ `ompi-info` (OpenMPI, fornece todas as configurações do MPI)
- ☐ `ldd` (bibliotecas compartilhadas da aplicação)
- ☐ `env` (variáveis de ambiente)
- ☐ `nm` (símbolos de um binário)



### 3. Registro automático de informações

Deve-se automatizar o registro de informações do sistema

- Coletadas toda vez que um experimento for realizado
- Armazenadas juntamente com os resultados do experimento

#### HW (Plataforma)

- ☐ `lstopo`, da ferramenta `hwloc` (topologia do sistema)
- ☐ `cpufreq-info` (frequência atual, mínima, máxima, governor)
- ☐ `ip` (ou `ifconfig`, obter configurações da interface de rede)
- ☐ `lspci` (todos os dispositivos PCI)

#### SW (Aplicação paralela)

- ☐ `mpi-info` (OpenMPI, fornece todas as configurações do MPI)
- ☐ `ldd` (bibliotecas compartilhadas da aplicação)
- ☐ `env` (variáveis de ambiente)
- ☐ `nm` (símbolos de um binário)

Outras informações que dependem do tipo do experimento.

## 4. Controle de Software (dependências da aplicação)

Aplicações paralelas **dependem de inúmeras bibliotecas**

- Solvers de álgebra linear (BLAS)
- Bibliotecas de comunicação (MPI, OpenMP)

Além disso, pode-se querer testar **múltiplas versões** das dependências

## 4. Controle de Software (dependências da aplicação)

Aplicações paralelas **dependem de inúmeras bibliotecas**

- Solvers de álgebra linear (BLAS)
- Bibliotecas de comunicação (MPI, OpenMP)

Além disso, pode-se querer testar **múltiplas versões** das dependências

**Spack** – <<https://github.com/spack/spack>>

- Gerenciador de pacotes em nível de usuário
- Muitas configurações do mesmo pacote podem coexistir
- Sintaxe específica para especificar versões e opções de configurações

## 4. Controle de Software (dependências da aplicação)

Aplicações paralelas **dependem de inúmeras bibliotecas**

- Solvers de álgebra linear (BLAS)
- Bibliotecas de comunicação (MPI, OpenMP)

Além disso, pode-se querer testar **múltiplas versões** das dependências

**Spack** – <<https://github.com/spack/spack>>

- Gerenciador de pacotes em nível de usuário
- Muitas configurações do mesmo pacote podem coexistir
- Sintaxe específica para especificar versões e opções de configurações

```
git clone https://github.com/spack/spack.git
cd spack/bin
./spack install zlib@1.2.8+pic~shared+optimize
```

# Um exemplo com libboost, MPI e gcc

spack spec mostra o que será instalado com a especificação fornecida

```
./spack spec boost@1.69.0+mpi^openmpi@2.0 %gcc@8.2
```

Input spec

```
-----  
boost@1.69.0+mpi  
  ^openmpi@2.0%gcc@8.2
```

Concretized

```
-----  
boost@1.69.0%gcc@8.2+atomic+chrono~clanglibcpp~context~coroutine cxxstd=98 +date_time~debug+exception~fiber+  
  filesystem+graph~icu+iostreams+locale+log+math+mpi+multithreaded~numpy patches=2  
  ab6c72d03dec6a4ae20220a9dfd5c8c572c5294252155b85c6874d97c323199,3  
  a83d907043708218325c35ffc318fd6d6cfd78ba89a78f2c70013c72603e5b8,607  
  b0772dec1287c9084ae3b36ee32bff945a2fe5e608823ed47a1ea765c84cd ~pic+program_options~python+random+regex+  
  serialization+shared+signals~singlethreaded+system~taggedlayout+test+thread+timer~versionedlayout+wave  
  arch=linux-debiantesting-x86_64  
^bzip2@1.0.6%gcc@8.2+shared arch=linux-debiantesting-x86_64  
  ^diffutils@3.7%gcc@8.2 arch=linux-debiantesting-x86_64  
^openmpi@2.0%gcc@8.2+cuda+cxx_exceptions fabrics=auto ~java~legacylaunchers~memchecker patches=  
  d26978ea058131ced4e51668a524f556d3d90d178d54634e6f2077f4c8ba7762 ~pmi schedulers=auto ~sqlite3~  
  thread_multiple+vt arch=linux-debiantesting-x86_64  
^hwloc@1.11.11%gcc@8.2~cairo~cuda~gl+libxml2~nvml+pci+shared arch=linux-debiantesting-x86_64  
  ^libpciaccess@0.13.5%gcc@8.2 arch=linux-debiantesting-x86_64  
    ^libtool@2.4.6%gcc@8.2 arch=linux-debiantesting-x86_64
```

## 5. Controle em nível de Sistema Operacional (SO)

Spack é tri, mas não permite controlar toda a pilha de software

- O SO tem uma influência por vezes determinando na variabilidade

## 5. Controle em nível de Sistema Operacional (SO)

Spack é tri, mas não permite controlar toda a pilha de software

- O SO tem uma influência por vezes determinando na variabilidade

Método Nativo → Kadeploy3

<<http://kadeploy3.gforge.inria.fr/>>

- Gerencia e utiliza perfis PXE com servidor TFTP
- Dispara comandos de reboot com IPMI ou através de PDU gerenciável
- O usuário instala seu SO nativamente em uma partição do disco

## 5. Controle em nível de Sistema Operacional (SO)

Spack é tri, mas não permite controlar toda a pilha de software

- O SO tem uma influência por vezes determinando na variabilidade

Método Nativo → Kadeploy3

<<http://kadeploy3.gforge.inria.fr/>>

- Gerencia e utiliza perfis PXE com servidor TFTP
- Dispara comandos de reboot com IPMI ou através de PDU gerenciável
- O usuário instala seu SO nativamente em uma partição do disco

Método Virtualizado → CharlieCloud

<<https://github.com/hpc/charliecloud>>

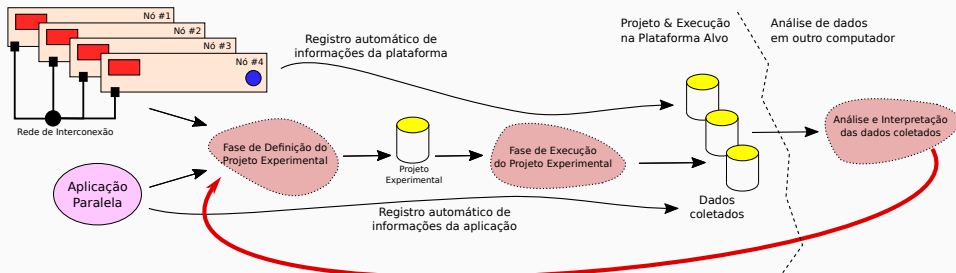
- Baseado com Linux Containers
- Não exige hardware específico (apenas o suporte à virtualização)



# Análise de Dados

---

## Resultado dos experimentos: conjunto de arquivos (*checkpoint*)



Análise é conduzida *offline*, após a execução do experimento

- No computador pessoal do investigador
- Plataforma diferente daquela usada nos experimentos

Processo iterativo de análise de dados

# Adotar uma estratégia sistematizada de análise

Permitir

- Reexecutar algumas etapas do processo de análise
- Revisar o fluxo de transformações de dados

## Permitir

- Reexecutar algumas etapas do processo de análise
- Revisar o fluxo de transformações de dados
- “Lembrar” como uma figura foi concebida (transformação e dados)
- Saber precisamente como se chegou a valores relatados no artigo

# Adotar uma estratégia sistematizada de análise

## Permitir

- Reexecutar algumas etapas do processo de análise
- Revisar o fluxo de transformações de dados
- “Lembrar” como uma figura foi concebida (transformação e dados)
- Saber precisamente como se chegou a valores relatados no artigo

Empregar ferramentas modernas de  
Ciência de Dados para uma Análise  
Reprodutível



# Adotar uma estratégia sistematizada de análise

Permitir

- Reexecutar algumas etapas do processo de análise
- Revisar o fluxo de transformações de dados
- “Lembrar” como uma figura foi concebida (transformação e dados)
- Saber precisamente como se chegou a valores relatados no artigo

Empregar ferramentas modernas de  
Ciência de Dados para uma Análise  
Reprodutível



Proposta por Donald Knuth

Permite converter um documento fonte em duas representações distintas

- Um formato legível para humanos
- Outro apto para execução em computadores

Proposta por Donald Knuth

Permite converter um documento fonte em duas representações distintas

- Um formato legível para humanos
- Outro apto para execução em computadores

Na prática: é um arquivo com código e texto interpostos

---

Análise de resultados experimentais

- Permite manter em um mesmo documento
  - Anotações preliminares
  - Expectativas, suposições e reflexões
  - Comandos de análise
  - Visualização de resultados



# Programação literária com OrgMode

OrgMode, uma extensão do editor de texto Emacs



Arquivos `.org`

- Blocos de código (em diversas línguas) com o pacote `Babel`
- Possível de exportar para `tex`, `pdf`, `html`, `odt`, ...

Caderno de Anotações

LabBook.org

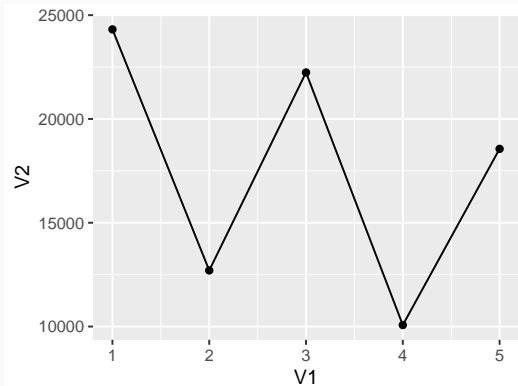
## Uma pequena demonstração com Org falando shell

```
export RANDOM=0
for n in `seq 5`;
do
    printf "%d $RANDOM \n" $n ;
done
```

```
1 24315
2 12703
3 22240
4 10073
5 18561
```

## Uma pequena demonstração com Org falando R

```
library(tidyverse)
dados %>%
  ggplot(aes(V1, V2)) +
  geom_point() + geom_line()
```



# Checklist para gráficos de qualidade

Ler documento auxiliar (texto do minicurso), mas vejamos um trecho...

Dados	✓	O tipo do gráfico é adequado para a natureza do dado (curva, barras, setores, histograma, nuvem de pontos, etc)
	✓	As aproximações/interpolações fazem sentido
	✓	As curvas são definidas com um número suficiente de pontos
	✓	O método de construção da curva é claro: interpolação (linear, polinomial, regressão, etc)
	✓	Os intervalos de confiança são visualizados (ou informados separadamente)
	✓	Os passos do histograma são adequados
	✓	Histogramas visualizam probabilidades (de 0 a 1)
Objetos Gráficos	✓	Os objetos gráficos são legíveis na tela, na versão impressa (P&B), em vídeo, etc
	✓	O intervalo do gráfico é padrão, sem cores muito similares, sem verde (vídeo)
	✓	Os eixos do gráfico estão claramente identificados e rotulados
	✓	Escalas e unidades estão explícitas
	✓	As curvas se cruzam sem ambiguidade
	✓	As grades ajudam o leitor
Anotações	✓	Eixos são rotulados por quantidades
	✓	Rótulos dos eixos são claros e autocontidos
	✓	Unidades estão indicadas nos eixos
	✓	Eixos são orientados da esquerda para a direita e de baixo para cima
	✓	Origem é (0,0), caso contrário deve estar claramente justificada
	✓	Sem buracos nos eixos
Anotações (2)	✓	Para gráficos de barras/histogramas a ordem das barras segue a ordenação clássica (alfabética, temporal, do melhor pro pior)
	✓	Cada curva tem uma legenda
	✓	Cada barra tem uma legenda
Informação	✓	Curvas estão na mesma escala

# Como fica um gráfico melhorado

```
library(tidyverse)

dados %>%
  ggplot(aes(V1, V2)) +
  theme_bw() +
  geom_point(size=2) +
  geom_line() +
  ylab("Valor Aleatório") +
  xlab("Observação") +
  ggtitle("Geração de Números
          Aleatórios em shell
          script") +
  lims(y = c(0, NA),
        x = c(1, NA))
```



# Reprodutibilidade da análise de desempenho

Ir além do texto científico (artigo, relatório) → Companion

O companion deve ter:

Caderno de anotações

Dados brutos

Código fonte

Dados processados

# Reprodutibilidade da análise de desempenho

Ir além do texto científico (artigo, relatório) → Companion

O companion deve ter:

Caderno de anotações	Dados brutos
Código fonte	Dados processados

## 1. Formato

- Dados: aberto e de estrutura simples → arquivos CSV
- Texto: usando programação literária (OrgMode, RMD, IPython/Jupyter)

## 2. Disponibilização

- Dificuldade de encontrar locais apropriados
  - Dados muito volumosos
- Alternativas
  - GitHub, Bitbucket, GitLab (restrições de volume)
  - FigShare, [Zenodo](#) (Ex. com DOI: <https://dx.doi.org/10.5281/zenodo.2605464>)

# Demonstração

---



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

1. Instalação de Ferramentas com Spack
2. Realização de Experimentos Computacionais
3. Análise de Dados
4. Criação de Gráficos

## Conclusão

---

## Terminologia da Association for Computing Machinery (ACM)

- Repetibilidade (mesmo time, mesma configuração experimental)
- Replicabilidade (time diferente, mesma configuração experimental)
- Reprodutibilidade (time diferente, configuração experimental diferente)

### Reflexão

Como validar/refutar resultados com um HW diferente?

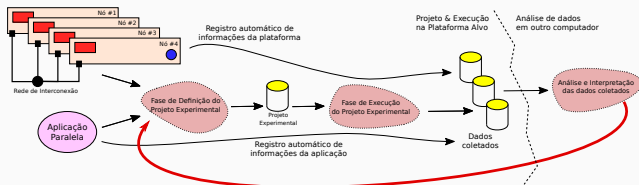
## Terminologia da Association for Computing Machinery (ACM)

- Repetibilidade (mesmo time, mesma configuração experimental)
- Replicabilidade (time diferente, mesma configuração experimental)
- Reprodutibilidade (time diferente, configuração experimental diferente)

## Reflexão

Como validar/refutar resultados com um HW diferente?

## melhorar nossas práticas experimentais



# 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), 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).



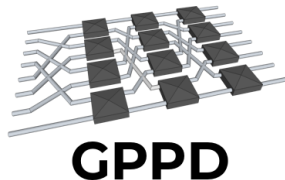
# Obrigado por participar e pela atenção! Perguntas?

## Contato

[vinicius.pinto@inf.ufrgs.br](mailto:vinicius.pinto@inf.ufrgs.br)

[lucas.nesi@inf.ufrgs.br](mailto:lucas.nesi@inf.ufrgs.br)

[schnorr@inf.ufrgs.br](mailto:schnorr@inf.ufrgs.br)





Este documento está licenciado sob a Licença *Atribuição-Compartilhual 4.0 Internacional (CC BY-SA 4.0)* da *Creative Commons* (CC). Em resumo, você deve creditar a obra da forma especificada pelo autor ou licenciante (mas não de maneira que sugira que estes concedem qualquer aval a você ou ao seu uso da obra). Você pode usar esta obra para fins comerciais. Se você alterar, transformar ou criar com base nesta obra, você poderá distribuir a obra resultante apenas sob a mesma licença, ou sob uma licença similar à presente. Para ver uma cópia desta licença, visite <<https://creativecommons.org/licenses/by-sa/4.0/>>.

Este documento foi produzido usando exclusivamente software livre: Sistema Operacional GNU/Linux, compilador de texto  $\text{\LaTeX}$ , editor gráfico Inkscape, *workflow* reproduzível em OrgMode com Emacs, as linguagens de programação R, com os pacotes do universo *tidyverse*, e *shell script*, o processador PS/PDF GhostScript, entre outros.