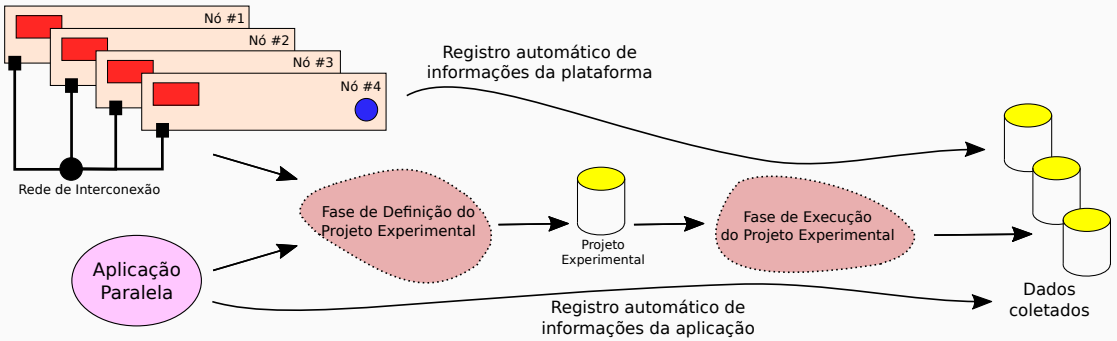
Boas Práticas para Experimentos Computacionais em Clusters

Projeto Experimental para Prática com Alto Desempenho

Lucas Mello Schnorr, Vinícius Garcia Pinto

19ª Escola Regional de Alto Desempenho da Região Sul (ERAD/RS)
Três de Maio – RS – Brasil
11 de abril de 2019

10ª Escola Regional de Alto Desempenho (ERAD/SP)
Campinas - SP - Brasil
13 de abril de 2019



Fonte da imagem: Fabricação própria com Inkscape

Apresentação

Para começar...

1. Baixar e lançar o container (dados são efêmeros)

```
docker pull schnorr/erad19
docker run -it schnorr/erad19
```

2. Configurar a chave privada para acesso ao cluster

```
wget http://www.inf.ufrgs.br/~schnorr/chave.rsa
chmod 600 chave.rsa
```

3. Confirmar que se consegue acessar o parque do grupo

```
ssh -i chave.rsa erad@gppd-hpc.inf.ufrgs.br
```

4. Veja se tem alguém com alguma reserva (com slurm)

```
squeue
```

5. Copiar de/para o container (para conhecimento somente)

```
docker ps
docker cp foo.txt 72ca2488b353:/home/user
docker cp 72ca2488b353:/home/user/foo.txt .
```

Contextualização do Minicurso

Grande maioria dos trabalhos envolve experimentos computacionais

- Utilizam pelo menos uma máquina
- Podem envolver vários computadores

Contextualização do Minicurso

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

Contextualização do Minicurso

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 portáveis 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

Objetivos

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)

Estrutura do Minicuro

ERAD/RS 2019, MC #5 (Lab. #6)

Sessão #1: 11/04, 9:30h – 10:30h (Introdução, Controle e Coleta, Análise de Dados)

Sessão #2: 11/04, 14h – 16h (Demonstração, Prática, Fechamento)

- Instal. ferramentas (SPACK)
- Exp. computacionais
  - Reserva de nós (SLURM)
  - Coleta de dados (bash)
  - Execução de aplicação paralela
- Análise de dados (R+tidyverse)
- Criação de gráficos (ggplot2)

ERAD/SP 2019, Tutorial II (Sala 303)

Sessão #1: 13/04, 15h – 16:30h (Introdução, Controle e Coleta, Análise de Dados)

Sessão #2: 13/04, 17h – 18h30 (Demonstração, Prática, Fechamento)

- Instal. ferramentas (SPACK)
- Exp. computacionais
  - Reserva de nós (SLURM)
  - Coleta de dados (bash)
  - Execução de aplicação paralela
- Análise de dados (R+tidyverse)
- Criação de gráficos (ggplot2)

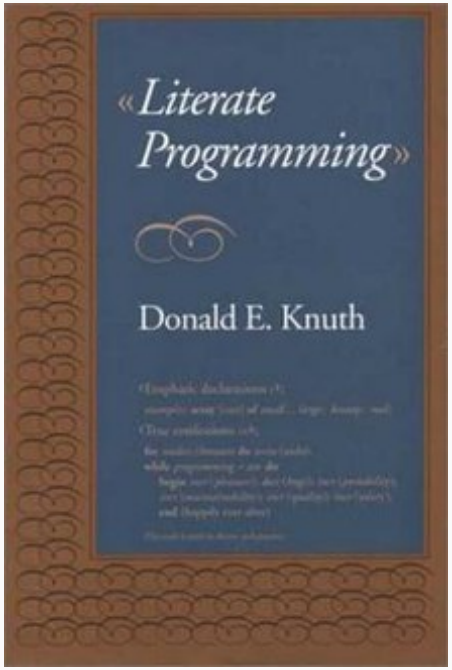
Referência

Literate Programming. Donald E. Knuth. (#1)

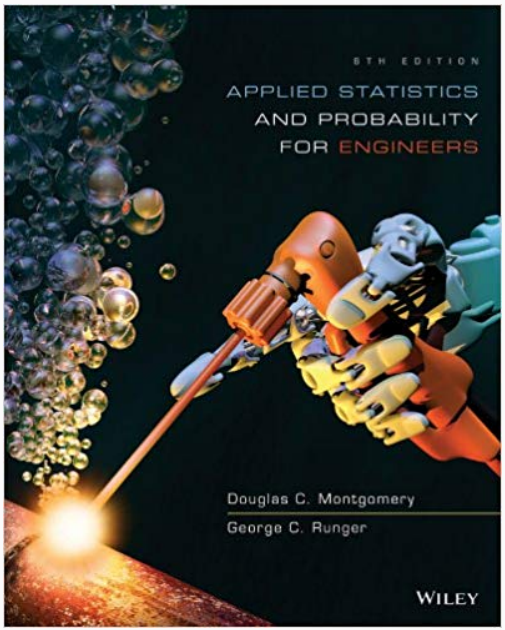
Applied Statistics and Probability for Engineers. Montgomery & Runger. (#2)

R for Data Science. Grolemund & Wickham. (#3)

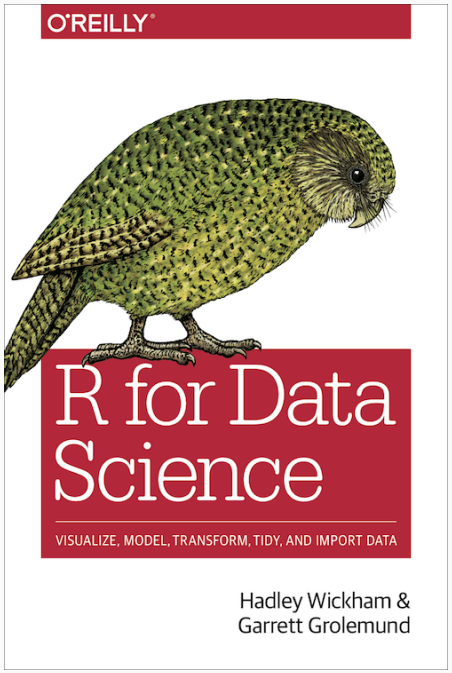
#1



#2



#3



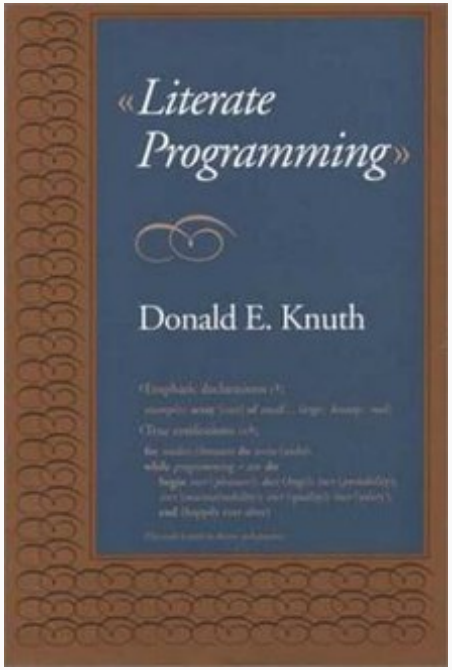
Referência

Literate Programming. Donald E. Knuth. (#1)

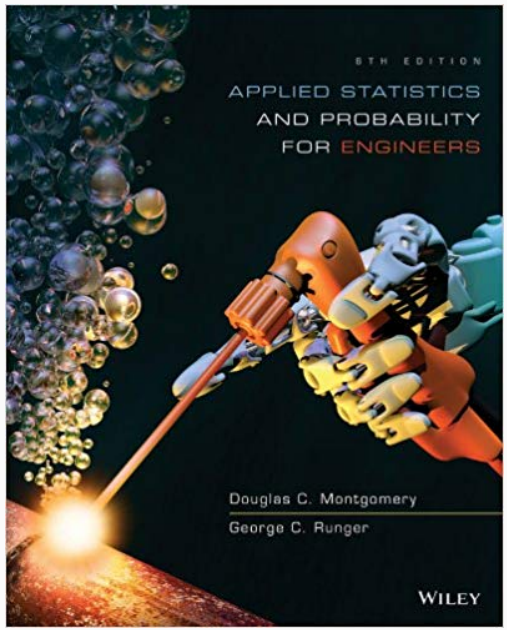
Applied Statistics and Probability for Engineers. Montgomery & Runger. (#2)

R for Data Science. Grolemund & Wickham. (#3)

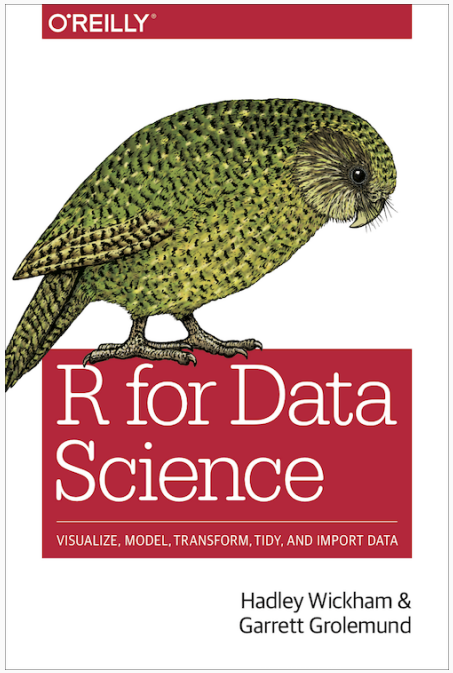
#1



#2



#3



Exemplos oriundos dos trabalhos do grupo de pesquisa

Propaganda

Instituto de Informática em Porto Alegre – RS

Vista geral de Porto Alegre em outubro, do Morro da Glória, 287 metros

Instituto de Informática em Porto Alegre – RS

Vista geral de Porto Alegre em outubro, do Morro da Glória, 287 metros

Vista do centro histórico de Porto Alegre, do nível do Guaíba

Instituto de Informática em Porto Alegre – RS

Vista geral de Porto Alegre em outubro, do Morro da Glória, 287 metros

Vist

Vista dos jardins do Instituto de Informática, manhã fria de agosto

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 continuo

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 continuo

Fatos marcantes

- Conceito máximo (7) pela CAPES (melhor da região sul)
- Internacionalização da formação e da investigação

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 continuo

Fatos marcantes

- Conceito máximo (7) pela CAPES (melhor da região sul)
- Internacionalização da formação e da investigação

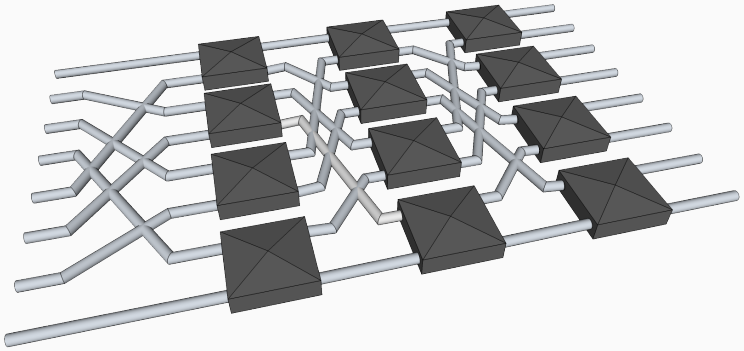
O evento ERAD/RS é uma boa oportunidade para sondar orientadores.

GPPD - Grupo de Processamento Paralelo e Distribuído

Site do grupo de pesquisa:

`http://www.inf.ufrgs.br/gppd/site/`

Logo do grupo de pesquisa



Eixos principais de investigação

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

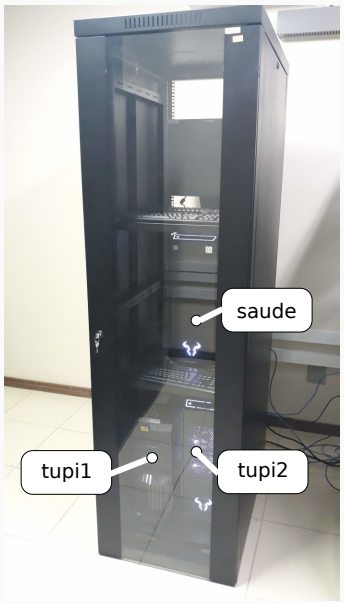
Parque Computacional de Alto Desempenho (PCAD)

Site: `http://gppd-hpc.inf.ufrgs.br/`

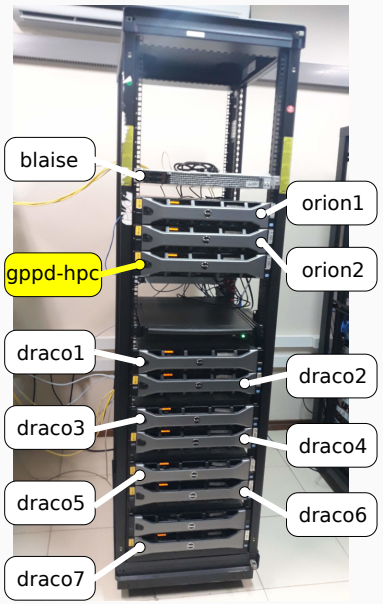
Possui aproximadamente 30 nós: 708 núcleos de CPU e 73.280 de GPU

- Computação de alto desempenho heterogênea

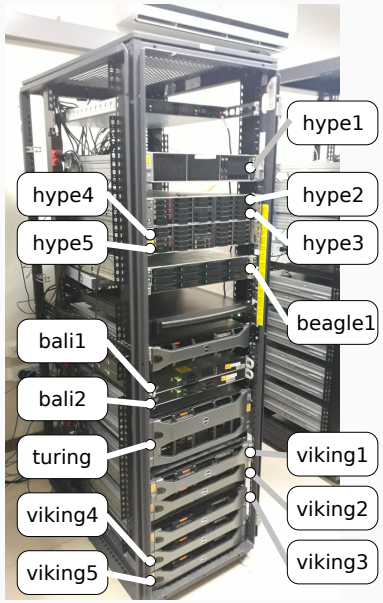
Rack #1



Rack #2



Rack #3



Temos um GT (Grupo de Trabalho)

- Formamos alunos no gerenciamento destas plataformas
- SLURM, NFS, LDAP, ...

Quem somos nós?

Lucas Mello Schnorr

Prof. INF/UFRGS & PPGC

http://www.inf.ufrgs.br/~schnorr

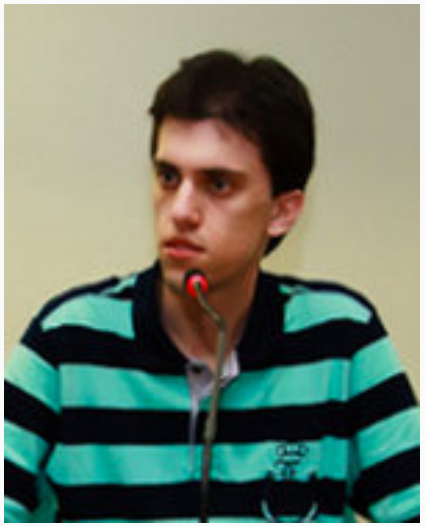
schnorr@inf.ufrgs.br

Vinícius Garcia Pinto

Prof. substituto do INF/UFRGS

http://www.inf.ufrgs.br/~vgpinto

vgpinto@inf.ufrgs.br



Introdução

Método Científico em Sistemas de Computação

Experimentos

validar ou refutar hipóteses

Confiabilidade $\rightarrow$ experimentos reprodutíveis

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

Método Científico em Sistemas de Computação

Experimentos

validar ou refutar hipóteses

Confiabilidade $\rightarrow$ experimentos reprodutíveis

1. Exercer um controle sobre as variáveis controláveis
2. Registrar o alor 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

Vantagens e desvantagens

Controle experimental

Desvantagens

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

Vantagens e desvantagens

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 e desvantagens

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

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

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
  - Redes
  - Programação
  - Análise de dados
  - Processamento Paralelo

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
  - Redes
  - Programação
  - Análise de dados
  - Processamento Paralelo
- Deve-se começar mesmo com uma estratégia simples

Visão geral

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

Visão geral

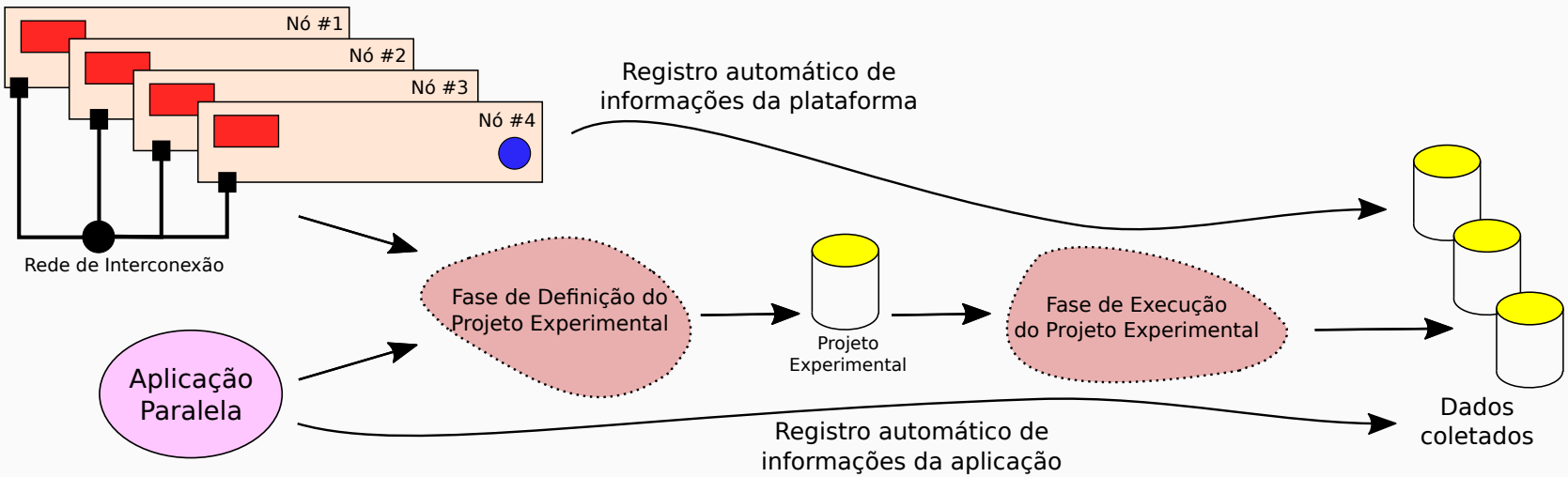
1. Teórica: Controle e Coleta
   - Lista não exaustiva de controle de sistemas computacionais
   - Projeto experimental
2. Teórica: Análise de Dados
   - Como analisar os dados com ferramentas modernas de *data science*?
   - Programação literária

Prática, com quatro tutoriais curtos

1. Instalação de ferramentas que permitem a rastreabilidade (`spack`)
2. Realização de experimentos computacionais (`slurm`, `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 $\rightarrow$ aumento da variabilidade

Fatores

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

Características impactantes $\rightarrow$ 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 $\rightarrow$ 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 $\rightarrow$ aumento da variabilidade

Fatores

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

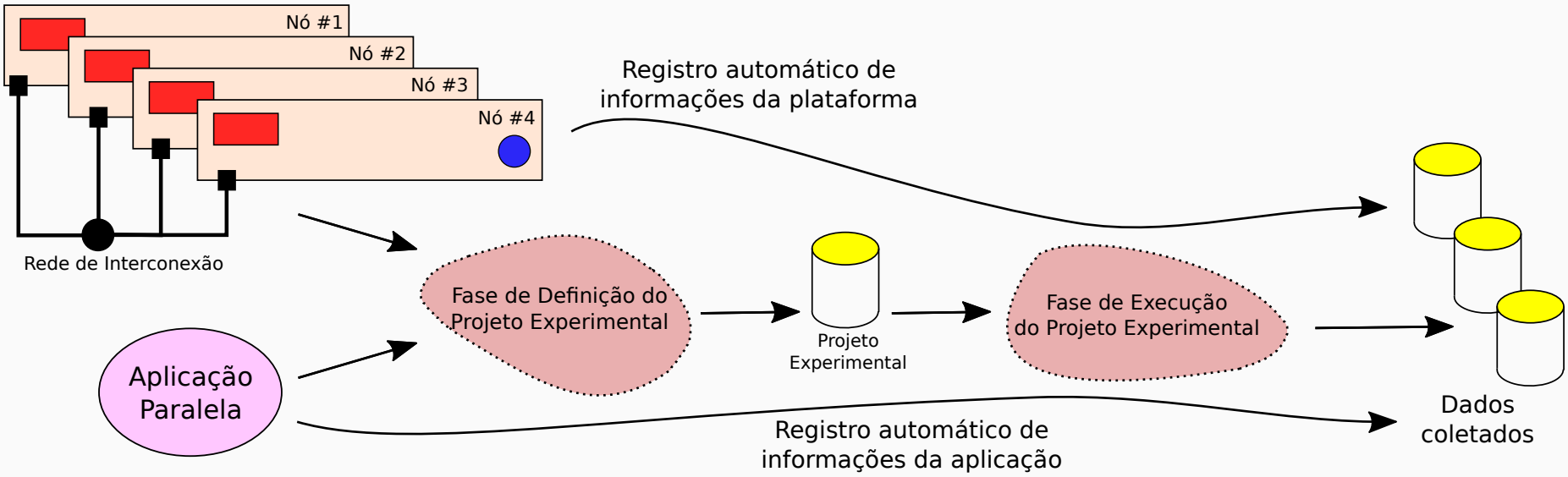
Aumento da Variabilidade dos experimentos

- 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
6. Integração com gerenciadores de *Jobs*

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

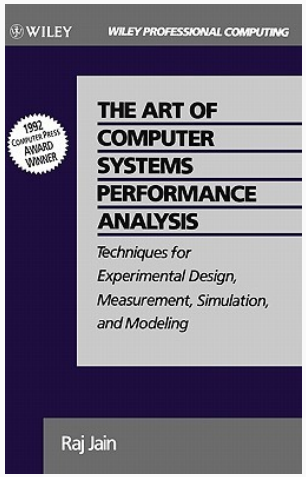
- Fatores (variáveis de controle) e níveis (seus valores)
- Variáveis de resposta (observações medidas)

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

```
// para cada processo em 1 2 4 8 16 32 64 128 256
//   para cada quantidade de nós 1 2 4 8
//      para cada frequência do processador 1.2 1.8 2.1
//         para cada rede 1GB 10GB 40GB
//            repita 10 vezes esta configuração
```

Tipos de projetos experimentais

Projetos simples, variam um único fator a cada vez

- Não permite o estudo da interação entre fatores

```
// para cada processo em 1 2 4 8 16 32 64 128 256
//   para cada quantidade de nós 1 2 4 8
//      para cada frequência do processador 1.2 1.8 2.1
//         para cada rede 1GB 10GB 40GB
//            repita 10 vezes esta configuração
```

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

```
// para cada processo em 1 2 4 8 16 32 64 128 256
//   para cada quantidade de nós 1 2 4 8
//      para cada frequência do processador 1.2 1.8 2.1
//         para cada rede 1GB 10GB 40GB
//            repita 10 vezes esta configuração
```

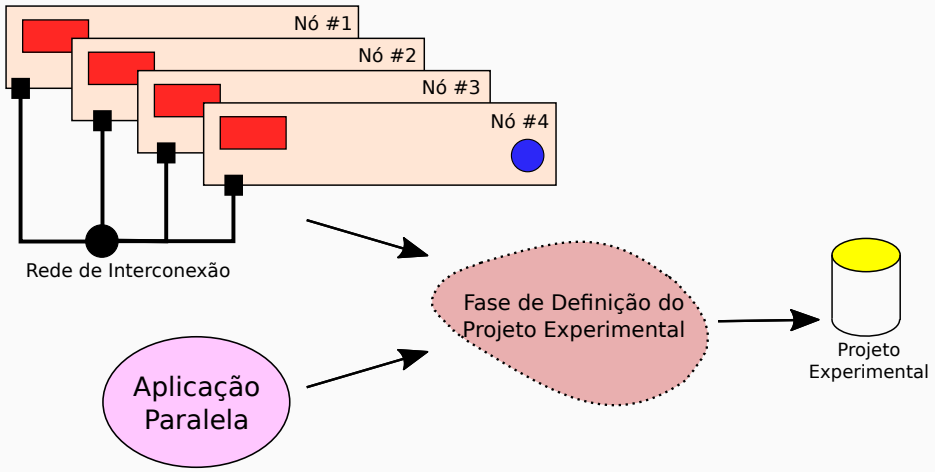
Projeto fatorial completo, todas as combinações possíveis de fatores

- 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

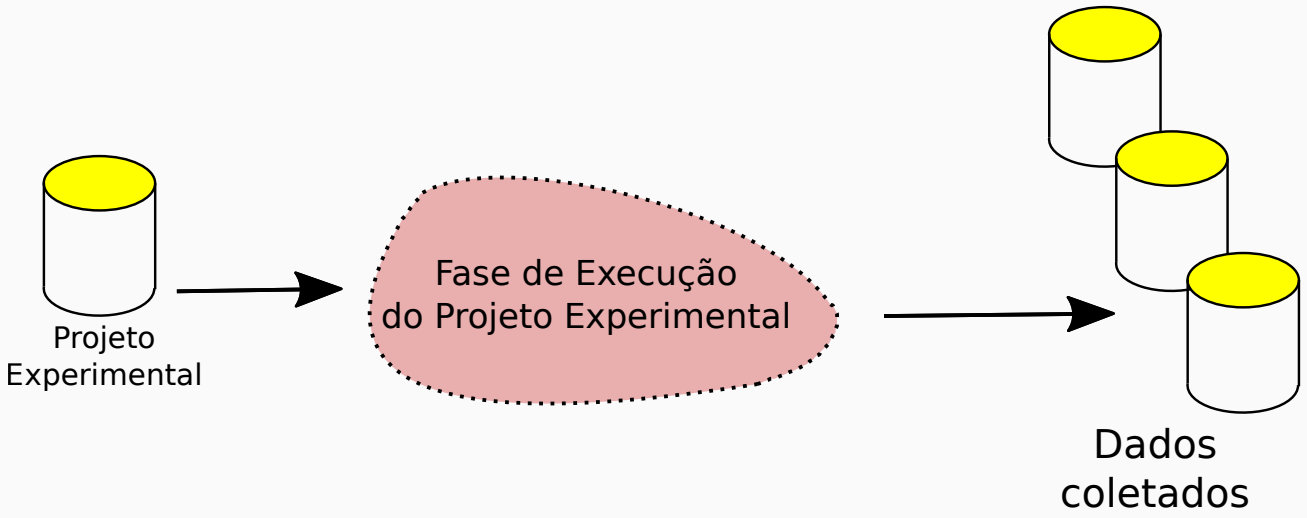
O projeto é registrado em um arquivo CSV

- 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 $\rightarrow$ quais configurações fixar? Depende.

- Por que fazer `drop caches` quando a aplicação não faz IO?
- Por que desativar *hyperthreading* se há um processo por nó?

2. Controle da complexidade

Diminuir a variabilidade $\rightarrow$ quais configurações fixar? Depende.

- Por que fazer `drop caches` quando a aplicação não faz IO?
- Por que desativar *hyperthreading* se há um processo por nó?

Listagem não exaustiva

- $\square$ 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 $\rightarrow$ quais configurações fixar? Depende.

- Por que fazer `drop caches` quando a aplicação não faz IO?
- Por que desativar *hyperthreading* se há um processo por nó?

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*)
  - Desativar *turboboost* (em processadores Intel)
    - Não há como saber quando é ativado ou desativado

2. Controle da complexidade

Diminuir a variabilidade $\rightarrow$ quais configurações fixar? Depende.

- Por que fazer `drop caches` quando a aplicação não faz IO?
- Por que desativar *hyperthreading* se há um processo por nó?

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*)
  - Desativar *turboboost* (em processadores Intel)
    - Não há como saber quando é ativado ou desativado
- □ Desativar *hyperthreading* (em processadores 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.

- Por que fazer `drop caches` quando a aplicação não faz IO?
- Por que desativar *hyperthreading* se há um processo por nó?

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*)
  - Desativar *turboboost* (em processadores Intel)
    - Não há como saber quando é ativado ou desativado
- ☐ Desativar *hyperthreading* (em processadores 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: Sistema operacional, topologia de hardware e freq. do processador

- ☐ `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: Sistema operacional, topologia de hardware e freq. do processador

- ☐ `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: Informações associadas à aplicação paralela

- ☐ `ompi-info` (OpenMPI, fornece todas as configurações do MPI)
- ☐ `ldd` (mostra sa bibliotecas compartilhadas da aplicação)
- ☐ `env` (todas as variáveis de ambiente)
- ☐ `nm` (todos os 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: Sistema operacional, topologia de hardware e freq. do processador

- ☐ `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: Informações associadas à aplicação paralela

- ☐ `ompi-info` (OpenMPI, fornece todas as configurações do MPI)
- ☐ `ldd` (mostra sa bibliotecas compartilhadas da aplicação)
- ☐ `env` (todas as variáveis de ambiente)
- ☐ `nm` (todos os 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
```

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
```

Alternativa mais antiga: `http://modules.sourceforge.net/`

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_ti
    ^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~mem
        ^hwloc@1.11.11%gcc@8.2~cairo~cuda~gl+libxml2~nvml+pci+shared arch=linux-deb
            ^libpciaccess@0.13.5%gcc@8.2 arch=linux-debiantesting-x86_64
                ^libtool@2.4.6%gcc@8.2 arch=linux-debiantesting-x86_64
                    ^m4@1.4.18%gcc@8.2 patches=3877ab548f88597ab2327a2230ee048d2d07
                        ^libsigsegv@2.11%gcc@8.2 arch=linux-debiantesting-x86_64
                ^pkgconf@1.6.0%gcc@8.2 arch=linux-debiantesting-x86_64
                ^util-macros@1.19.1%gcc@8.2 arch=linux-debiantesting-x86_64
            ^libxml2@2.9.8%gcc@8.2~python arch=linux-debiantesting-x86_64
                ^libiconv@1.15%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)

6. Integração com gerenciadores de *Jobs*

Gerenciador de *jobs* em clusters de computadores

- Permite a alocação e reserva de nós
- Ferramentas: Slurm (SDumont, . . .), OAR (Grid5000)

6. Integração com gerenciadores de *Jobs*

Gerenciador de *jobs* em clusters de computadores

- Permite a alocação e reserva de nós
- Ferramentas: Slurm (SDumont, . . .), OAR (Grid5000)

Exemplo utilizando a sintaxe do Slurm

```
#!/bin/bash
#SBATCH --job-name="EXP00"
#SBATCH --nodes=16
#SBATCH --time=02:00:00
#SBATCH --partition=gppd-hpc
#SBATCH --output=%x_%j.out
#SBATCH --error=%x_%j.err

# Lançamento da execução do projeto experimental
```

6. Integração com gerenciadores de *Jobs*

Gerenciador de *jobs* em clusters de computadores

- Permite a alocação e reserva de nós
- Ferramentas: Slurm (SDumont, . . .), OAR (Grid5000)

Exemplo utilizando a sintaxe do Slurm

```
#!/bin/bash
#SBATCH --job-name="EXP00"
#SBATCH --nodes=16
#SBATCH --time=02:00:00
#SBATCH --partition=gppd-hpc
#SBATCH --output=%x_%j.out
#SBATCH --error=%x_%j.err

# Lançamento da execução do projeto experimental
```

Integração de tudo o que foi visto até agora

deve ser realizado automaticamente pelo script

Integração (não exaustiva) com Slurm

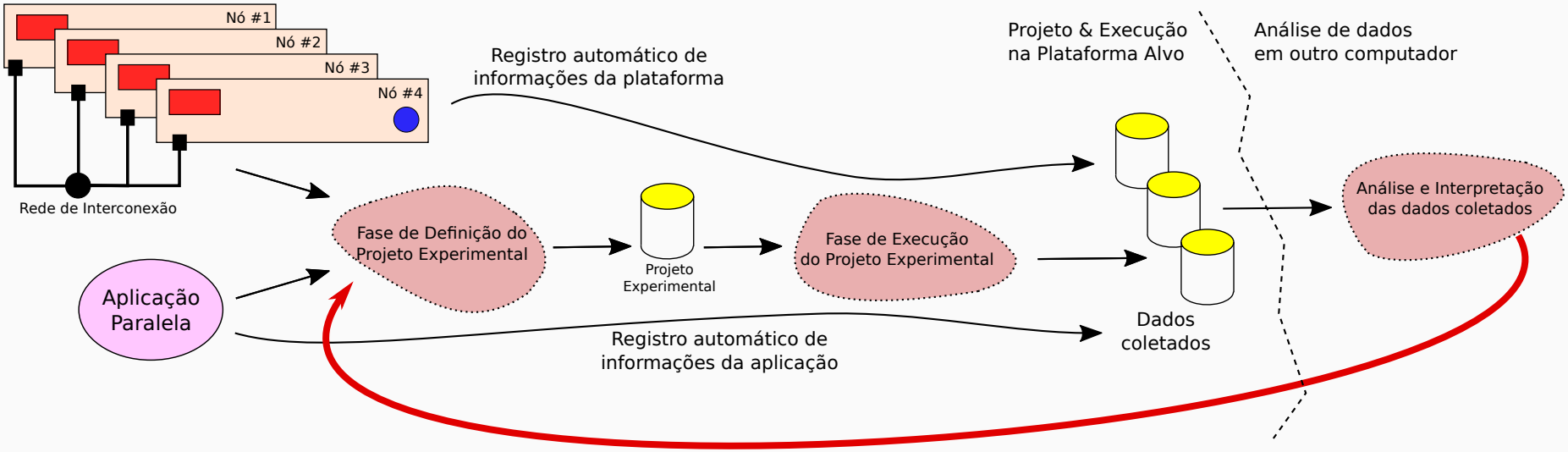
```
#!/bin/bash
#SBATCH --job-name="EXP00"
#SBATCH --nodes=16
#SBATCH --time=02:00:00
#SBATCH --partition=gppd-hpc
#SBATCH --output=%x_%j.out
#SBATCH --error=%x_%j.err

# 1. Controle inicial dos nós computacionais (HW e SW)
# 2. Registro das condições iniciais
# 3. Ler o projeto experimental, e para cada experimento
# 3.1 Aplicar os parâmetros (fatores e valores) específicos
# 3.2 Registro das condições iniciais do experimento
# 3.3 Executar o experimento
# 3.4 Coletar os dados do experimento em um diretório
# 4. Centralizar os dados em um único diretório
# 5. Arquivar este script e logs junto com os dados
```

Análise de Dados

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

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

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

Programação Literária + Ferramentas Modernas + Compartilhar

Programação Literária

Proposta por Donald Knuth

Permite converter um documento fonte em duas representações distintas

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

Programação Literária

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`, `odt`, ...

Caderno de Anotações

LabBook.org

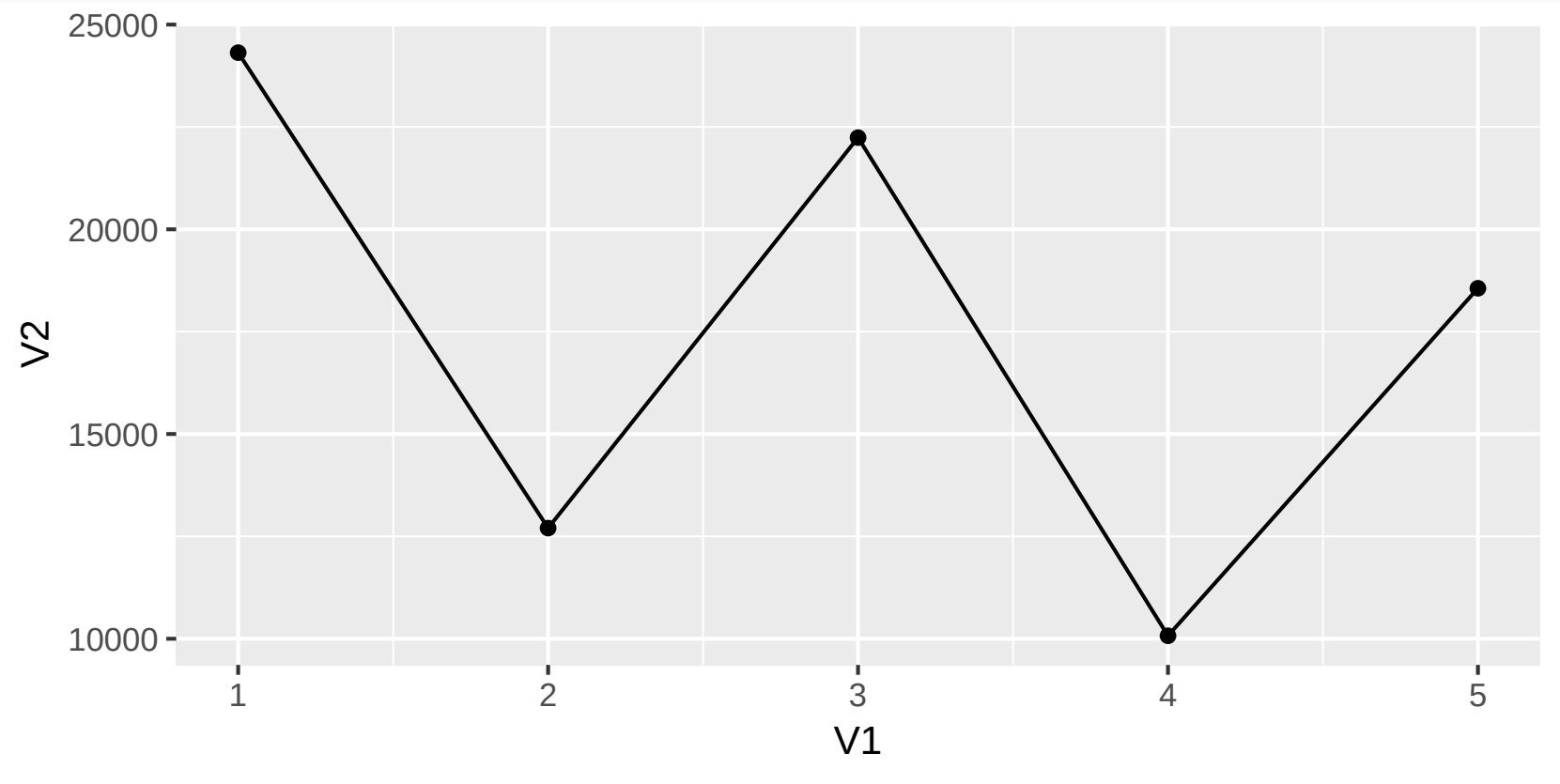
Uma pequena demonstração com Org falando `shell`

```
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()
```



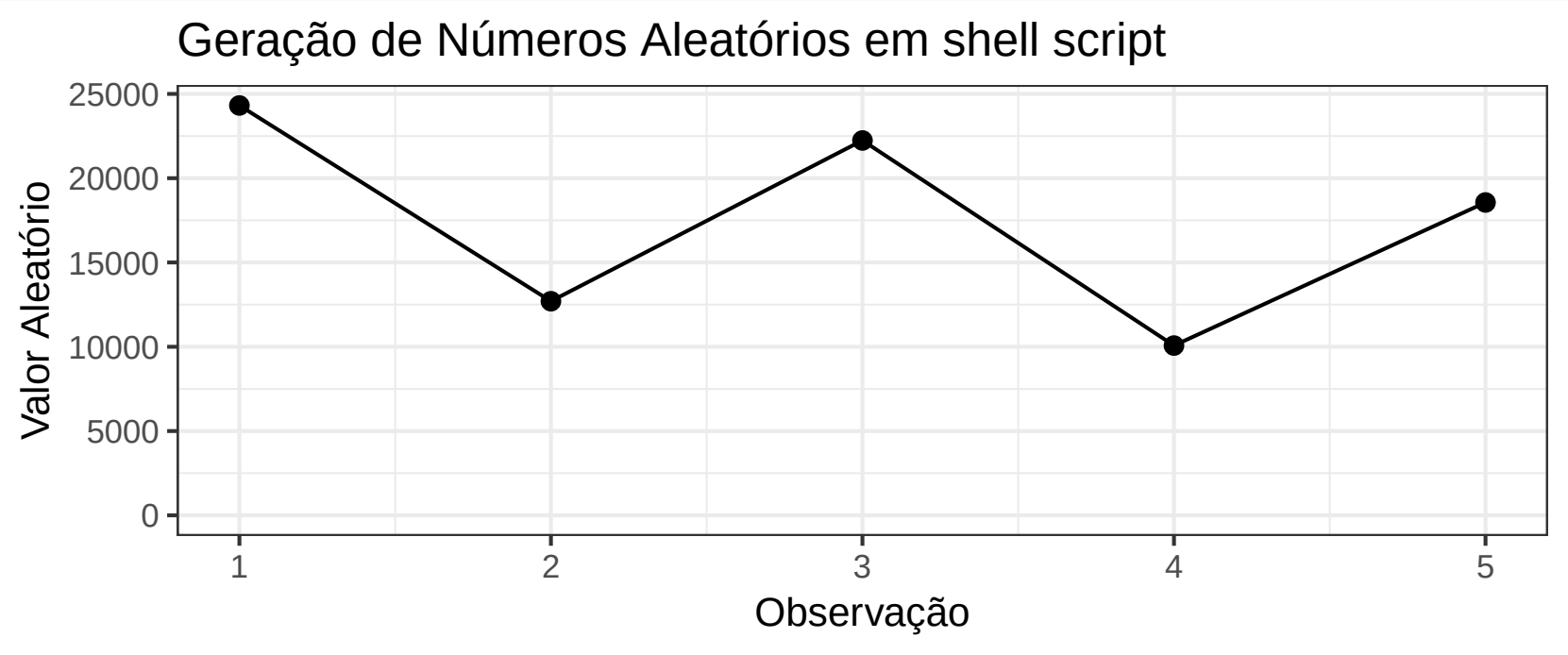
Checklist para gráficos de qualidade

Ler documento auxiliar, mas vejamos um trecho. . .

Dados	✓	O tipo do gráfico é adequado para a natureza do dad
	✓	As aproximações/interpolações fazem sentido
	✓	As curvas são definidas com um número suficiente de
	✓	O método de construção da curva é claro: interpolaçã
	✓	Os intervalos de confiança são visualizados (ou inform
	✓	Os passos do histograma são adequados
	✓	Histogramas visualizam probabilidades (de 0 a 1)
Objetos Gráficos	✓	Os objetos gráficos são legíveis na tela, na versão imp
	✓	O intervalo do gráfico é padrão, sem cores muito simil
	✓	Os eixos do gráfico estão claramente identificados e r
	✓	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 autocentidos

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) $\rightarrow$ Companion

O companion deve ter

- Caderno de anotações
- Código fonte
- Dados brutos
- Dados processados

Reprodutibilidade da análise de desempenho

Ir além do texto científico (artigo, relatório) $\rightarrow$ Companion

O companion deve ter

- Caderno de anotações
- Código fonte
- Dados brutos
- Dados processados

1. Formato
   - Dados: aberto e de estrutura simples $\rightarrow$ arquivos CSV
   - Texto: usando programação literária (OrgMode, RMD, IPython)
2. Disponibilização
   - Dificuldade de encontrar locais apropriados
     - Dados muito volumosos
   - Alternativas
     - GitHub, Bitbucket, GitLab (restrições de volume)
     - FigShare, Zenodo
   - Exemplo com DOI `10.5281/zenodo.2605464`
     `https://zenodo.org/record/2605464`

Demonstração

Tutorial

Tutorial (com acompanhamento do ministrante)

`https://github.com/viniciusvgp/tutorial-mc-erad-2019`

ou

`https://gitlab.com/schnorr/erad19`

1. Instalação de Ferramentas com Spack (local)
2. Realização de Experimentos Computacionais (remoto, no parque)

```
ssh erad@gppd-hpc.inf.ufrgs.br
```

3. Análise de Dados (local)
   `http://www.inf.ufrgs.br/~vgpinto/erad-tuto.tar`
4. Criação de Gráficos (local)

Conclusão

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?

Conclusão

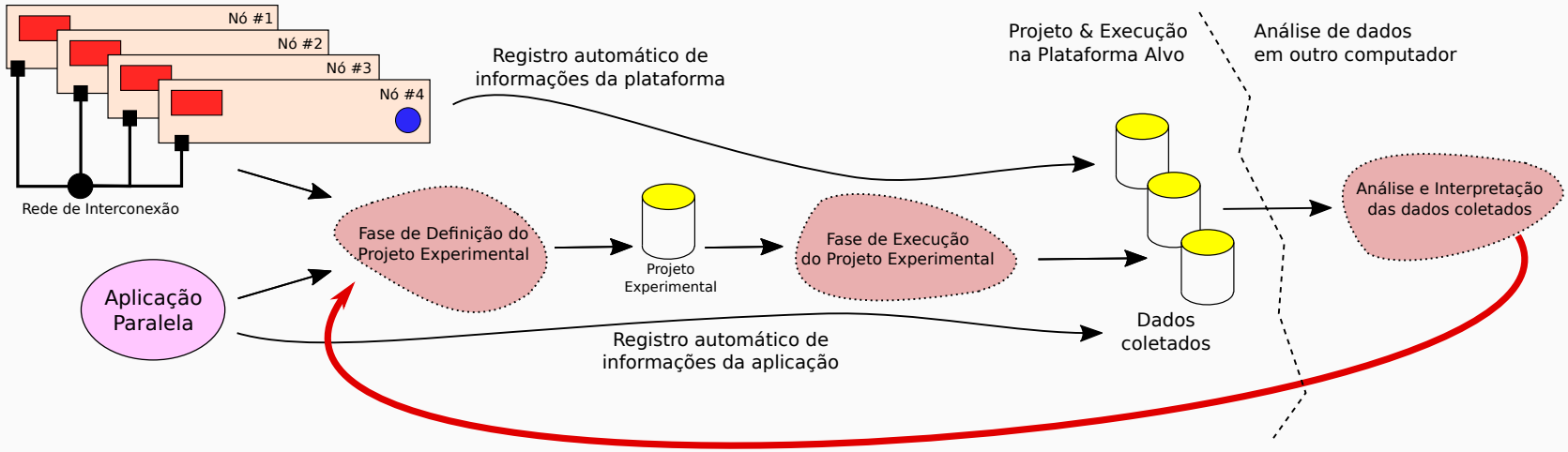
Terminologia da Association for Computing Machinery (ACM)

- 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?

melhorar nossas práticas experimentais



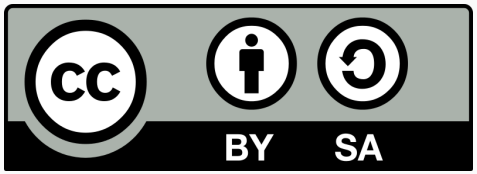
Obrigado por participar e pela atenção! Perguntas?

Contato

E-mail: schnorr@inf.ufrgs.br

Site: http://www.inf.ufrgs.br/~schnorr

Free Cultural Works



Este documento foi produzido usando exclusivamente software livre: Sistema Operacional Debian GNU/Linux, compilador de texto LaTeX, editor gráfico Inkscape, *workflow* reprodutível 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.