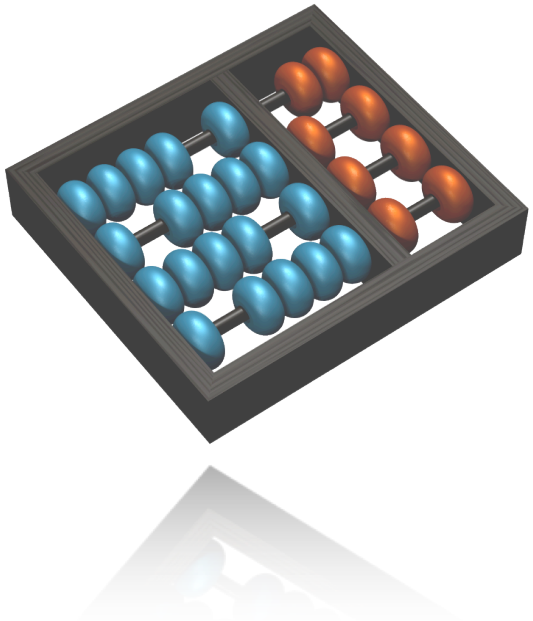
Técnicas para desenvolvimento e aceleração de códigos científicos



Prof. Edson Borin
edson@ic.unicamp.br
Instituto de Computação
UNICAMP

Minicurso
LNCC 2014

Sobre o minicurso

Segunda	Terça	Quarta	Quinta	Sexta
Introdução	Perfilamento - Contagem de tempo	Otimizações simples / compilação	SVN + CMake	GDB
Organização de processadores modernos	Otimização de acesso a dados	Bibliotecas otimizadas	Perfilamento - Detecção de código quente	Valgrind
Introdução ao laboratório		Vetorização de código		

Agenda

- Detecção de Código Quente
- Metodologia de Experimentação

Detecção de Código Quente

Código quente: domina o tempo de execução da aplicação.

Comece otimizando o código mais quente!

Senso comum: 10 / 90

90% do tempo é gasto em 10% do código

Detecção de Código Quente

Existem diversas ferramentas para detecção de código quente:

- gprof
- perf
- oprofile, Intel vtune, AMD codeanalist, Apple Instruments, ...

Detecção de Código Quente

Exemplo com o **gprof**:

Utilizar *flag* para instrumentar o código durante a compilação: (**-pg**)

```
gcc –pg teste.c –o teste
```

Executar a aplicação

```
./teste
```

Arquivo “gmon.out” é gerado

Detecção de Código Quente

Exemplo com o **gprof**:

```
gprof gmon.out
```

```
------------------------------------------------------------------
  %   cumulative  self              self     total
time   seconds   seconds   calls   ms/call  ms/call  name
70.45       5.14     5.14 26074562     0.00     0.00  intersect
26.01       7.03     1.90  4000000     0.00     0.00  shade
 3.72       7.30     0.27      100     2.71    73.03  calc_tile
------------------------------------------------------------------
```

Detecção de Código Quente

perf: disponível nas distribuições recentes do linux!

- Suporte do SO para leitura dos contadores de desempenho em HW!

Permite a amostragem de PC em função de eventos!!!

Detecção de Código Quente

Amostragem do PC no evento "**tempo**":

abra seu programa no gdb;

enquanto(o programa não terminar) {

  c // continue a execução

  ctrl+c // após algum **tempo** interrompa

  guarde o PC

}

faça um histograma das amostras

Detecção de Código Quente

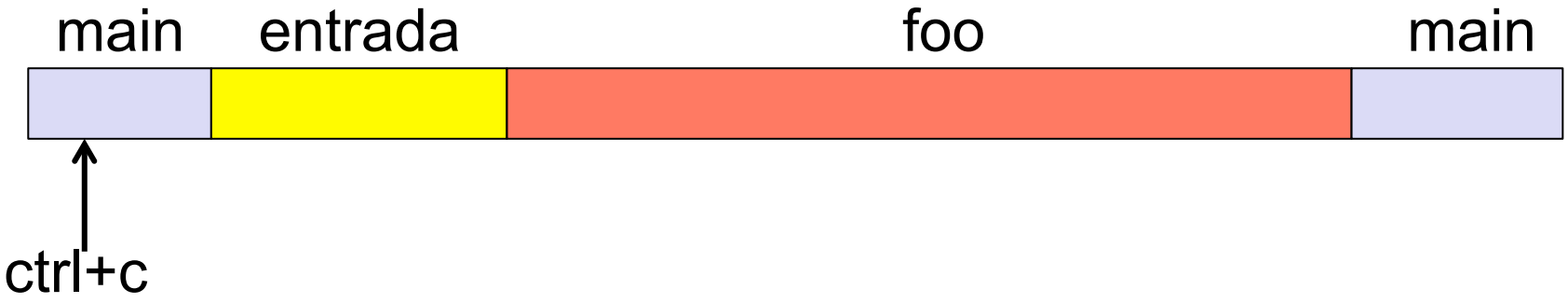
Amostragem do PC no evento "**tempo**":

Exemplo:

Detecção de Código Quente

Amostragem do PC no evento "**tempo**":

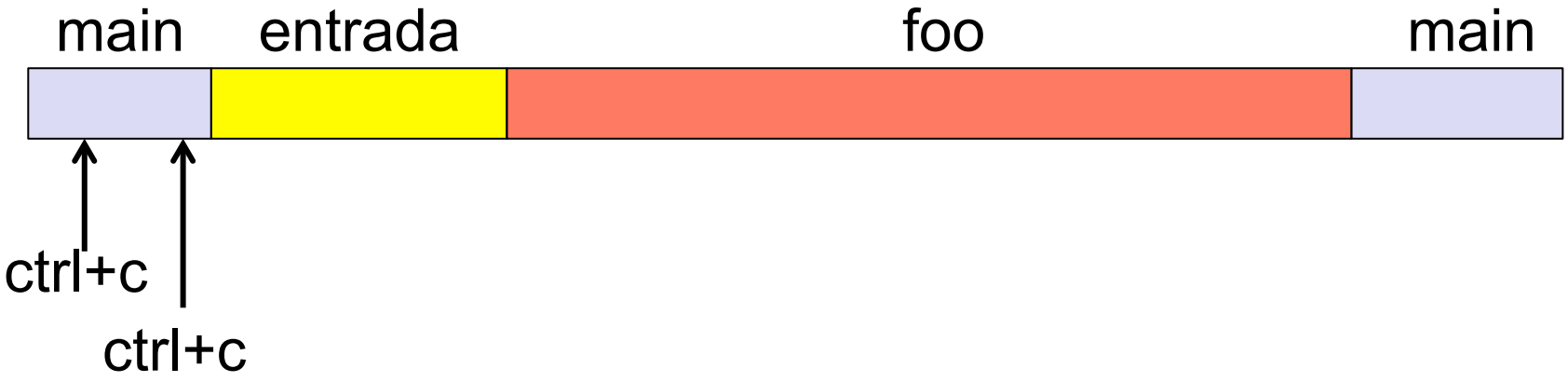
Exemplo:



Detecção de Código Quente

Amostragem do PC no evento "**tempo**":

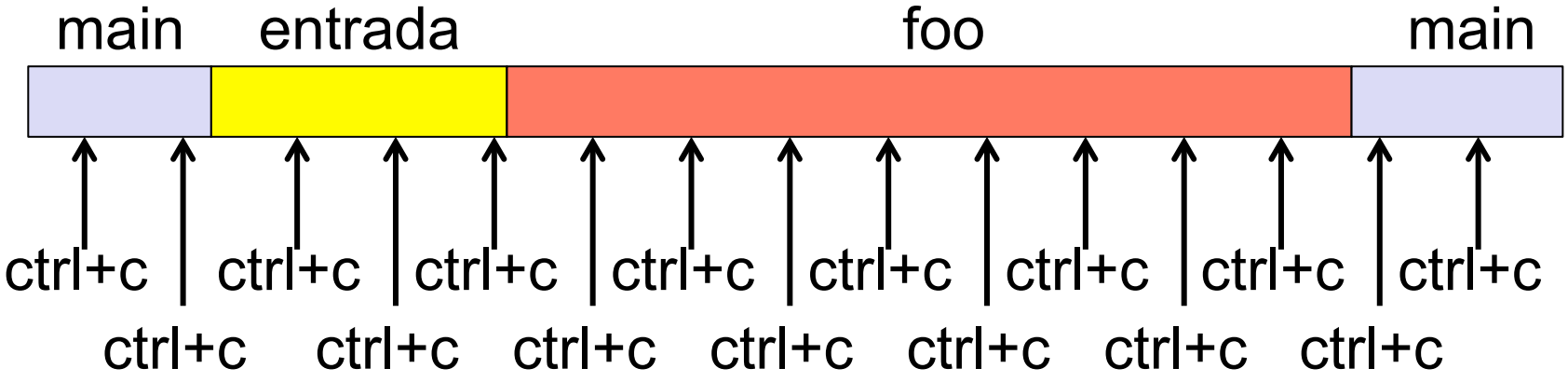
Exemplo:



Detecção de Código Quente

Amostragem do PC no evento "**tempo**":

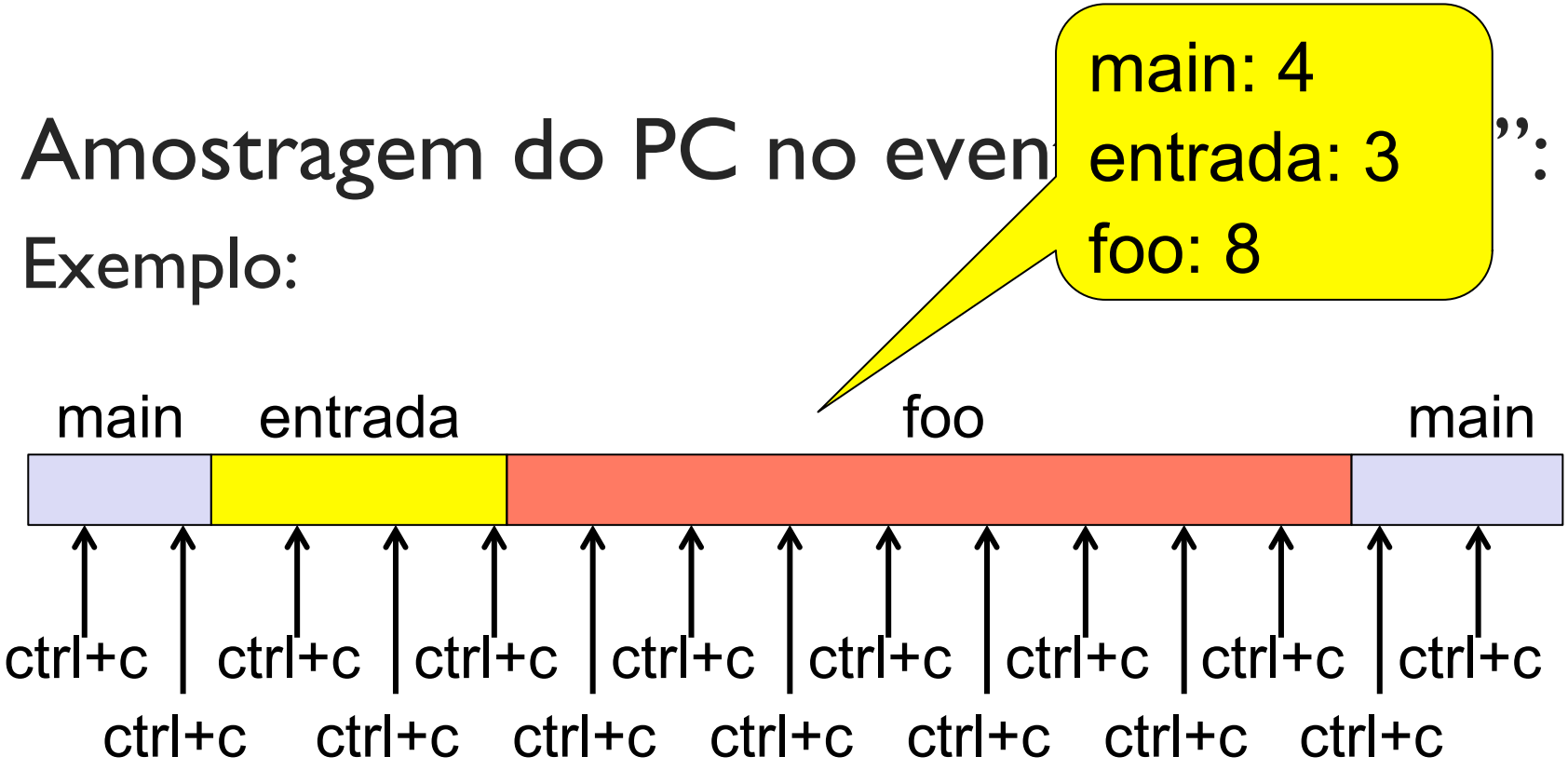
Exemplo:



Detecção de Código Quente

Amostragem do PC no even ”:

Exemplo:



Detecção de Código Quente

Amostragem do PC no evento "**tempo**"

GDB & ctrl+c não é muito prático

Utilize um programa de perfilamento

```
perf record ./my_prog.bin
perf report
```

Detecção de Código Quente

Amostragem do PC no evento "**tempo**"

perf report

```
# Events: 638  cycles
#
# Overhead        Command       Shared Object        Symbol
# ........  .............  ...................  ............
#
    70.76%  my_prog.bin    my_prog.bin          [.] clean_cache
    13.08%  my_prog.bin    my_prog.bin          [.] inner_prod
    11.59%  my_prog.bin    [kernel.kallsyms]    [k] 0xffffffff8103cf8a
     4.56%  my_prog.bin    my_prog.bin          [.] init_arrays
```

Detecção de Código Quente

Amostragem do PC no evento "**tempo**"

Se você quizer saber os trechos de código que tiveram o maior número de amostras:

```
perf record ./my_prog.bin
perf annotate
```

Detecção de Código Quente

```
 0.00 :400bc6: jmp    400bfa <inner_prod+0x48>
 8.54 :400bc8: mov    -0x4(%rbp),%eax
 1.22 :400bcb: cltq
 0.00 :400bcd: movsd  0x1472f0e0(,%rax,8),%xmm1
 4.88 :400bd6: mov    -0x4(%rbp),%eax
 3.66 :400bd9: cltq
 1.22 :400bdb: movsd  0x1372f0c0(,%rax,8),%xmm0
 0.00 :400be4: mulsd  %xmm1,%xmm0
 7.32 :400be8: movsd  -0x10(%rbp),%xmm1
62.20 :400bed: addsd  %xmm1,%xmm0
 0.00 :400bf1: movsd  %xmm0,-0x10(%rbp)
 2.44 :400bf6: addl   -bashx1,-0x4(%rbp)
 6.10 :400bfa: mov    -0x4(%rbp),%eax
 2.44 :400bfd: cmp    -bashx1fffff,%eax
 0.00 :400c02: jbe    400bc8 <inner_prod+0x16>
 0.00 :400c04: mov    -0x10(%rbp),%rax
```

Detecção de Código Quente

Outros eventos interessantes:

cache-misses, *cache-references*, *branch-misses*, etc...

```
perf –e cache-misses record ./app
perf annotate
```

Detecção de Código Quente

Outros eventos interessantes:

cache-misses, *cache-references*, *branch-misses*,

etc...

```
perf –e cache-misses record ./app
perf annotate
```

Amostragem! Ex: a cada 10.000 faltas na *cache* grava o PC!

Metodologia de Experimentação

Como medir e reportar o desempenho de aplicações?

Metodologia de Experimentação

Planejar e executar os experimentos.

Pontos importantes:

- Reprodutibilidade
- Controle de erros experimentais

Reprodutibilidade

Permite que o experimento e os resultados obtidos possam ser reproduzidos por outros.

- É importante para o avanço da ciência!
- Aumenta a credibilidade dos resultados e conclusões!

Reprodutibilidade

Permite que o experimento e os resultados obtidos possam ser reproduzidos por outros.

Para garantir a reprodutibilidade é necessário identificar e documentar os *fatores que podem afetar o resultado* do experimento.

Reprodutibilidade

Exemplos de fatores que podem afetar resultados:

Configuração do sistema:

HW: Tamanho das *caches*, tamanho e características da memória, tamanho da TLB, velocidade de E/S, ...

SW: Configuração do SO, ASLR, ...

Reprodutibilidade

Exemplos de fatores que podem afetar resultados:

Construção do SW:

Versão do compilador e regras usadas na compilação, bibliotecas, etc...

O código fonte utilizado no experimento!

Reprodutibilidade

Exemplos de fatores que podem afetar resultados:

Dados de entrada:

É importante e*specificar* e *disponibilizar* a base de dados e os argumentos utilizados nos experimentos.

Reprodutibilidade

Outros fatores que não são triviais:

- Ordem de ligação dos objetos.
- Quantidade e tamanho das variáveis de ambiente.

Mytkowicz, Hauswirth, Sweeney. Producing Wrong Data Without Doing Anything Obviously Wrong! ASPLOS'09

Reprodutibilidade

Parece trivial, mas poucos pesquisadores da área fazem de maneira apropriada!

> Progress in computational science is often hampered by researchers' inability to independently reproduce or verify published results. Attendees at a roundtable at

Yale Law School Round Table on Data and Code Sharing. Reproducible Research. Computing in Science & Engineering, 2010.

Reprodutibilidade

Parece trivial, mas poucos pesquisadores da área fazem de maneira apropriada!

> Have you ever tried to reproduce the results presented in a research paper? For many of our current publications, this would unfortunately be a challenging task. For a computational algorithm, details such as the exact data set,

Vandewalle et al. Reproducible research in signal processing. Signal Processing Magazine. IEEE, vol. 26, 2009.

Reprodutibilidade

Sugestões de leitura:

G. F. T. Gomes e Edson Borin. A Database for Reproducible Computational Research. WSCAD-SSC'12

Mytkowicz, Hauswirth e Sweeney. Producing Wrong Data Without Doing Anything Obviously Wrong! ASPLOS'09

Vandewalle et al. Reproducible research in signal processing. Signal Processing Magazine. IEEE, vol. 26, 2009.

Yale Law School Round Table on Data and Code Sharing. Reproducible Research. Computing in Science & Engineering, 2010.

Reprodutibilidade

Muitas vezes os resultados não podem ser reproduzidos **nem pelo próprio autor do experimento**!!!

Reprodutibilidade

Muitas vezes os resultados não podem ser reproduzidos **nem pelo próprio autor do experimento**!!!

Problema comum:

- sobrecarga do formalismo torna experimentação exploratória um processo lento.

Reprodutibilidade

Muitas vezes os resultados não podem ser reproduzidos **nem pelo próprio autor do experimento**!!!

Sugestão prática:

- Usar controle de versão para o *software*: SVN...
- Antes de reportar um resultado, fazer um *checkout* do código sem modificações locais e reproduzir o experimento. Reportar o resultado com o identificador da versão do SW.

Medindo Desempenho

Planejar e executar os experimentos.

Pontos importantes:

- Reprodutibilidade
- **Controle de erros experimentais**

Controle de Erros Experimentais

Dois tipos de erros:

a) erros aleatórios: resultados distorcidos em experimentos repetidos. A distorção é aleatória seguindo uma certa distribuição.

b) erros sistemáticos: resultados distorcidos em experimentos repetidos. A distorção é tendenciosa!

Controle de Erros Experimentais

Erros aleatórios:

Podem ser quantificados através da análise de múltiplas medidas. (desvio padrão, intervalo de confiança, ...)

Ex: Executar o experimento N vezes, computar a média e o intervalo de confiança.

Controle de Erros Experimentais

Erros aleatórios:

Podem ser reportados através de barras de erros nos gráficos ou mesmo no texto.

Controle de Erros Experimentais

Erros sistemáticos:

Difíceis de se detectar e não podem ser quantificados estatisticamente.

Causas comuns: variáveis ambiente, estruturas de *caching* do SO, processos concorrendo por recursos, serviços periódicos do SO.

Temperatura ambiente?

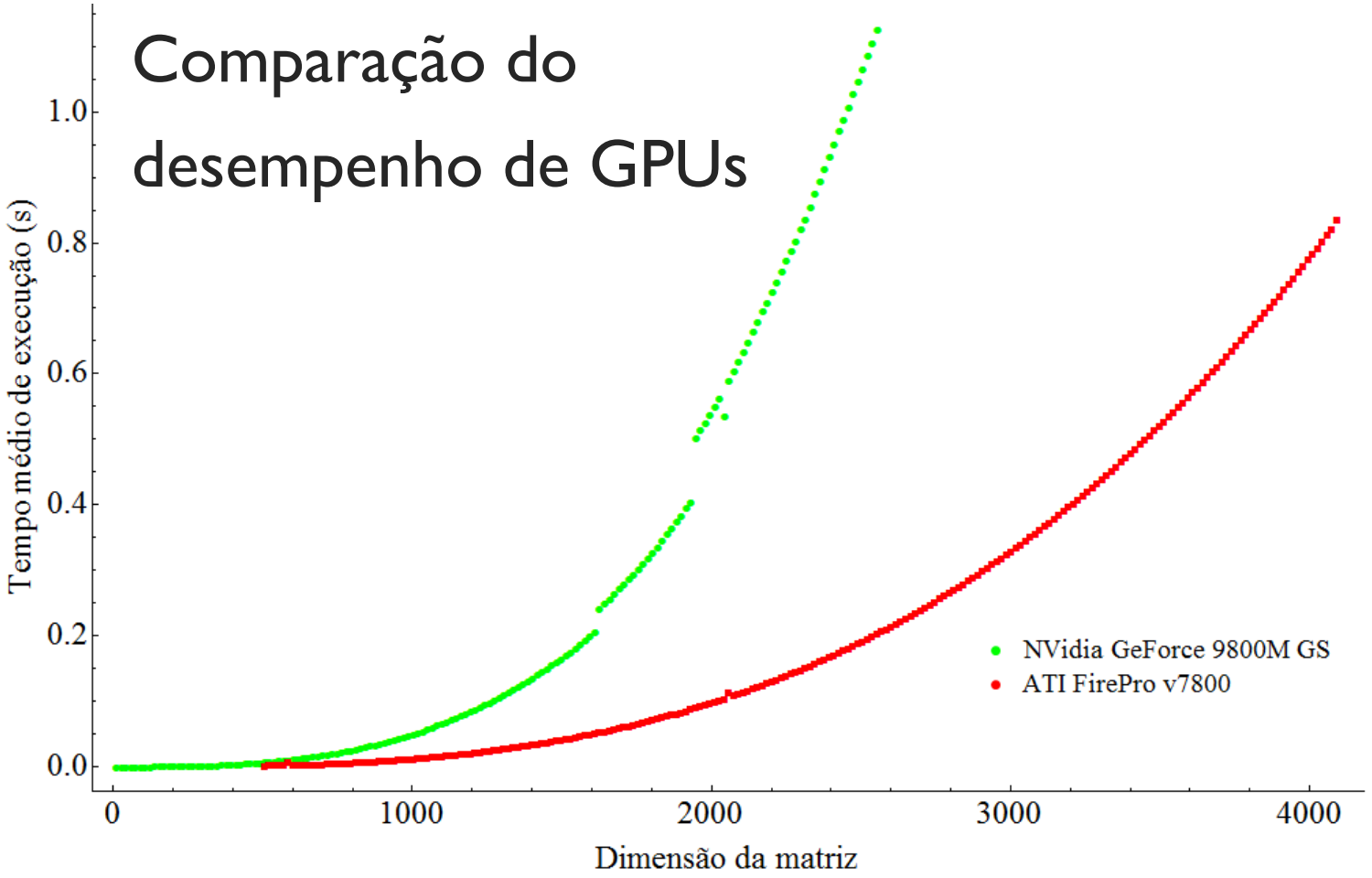
Controle de Erros Experimentais

Erros sistemáticos:

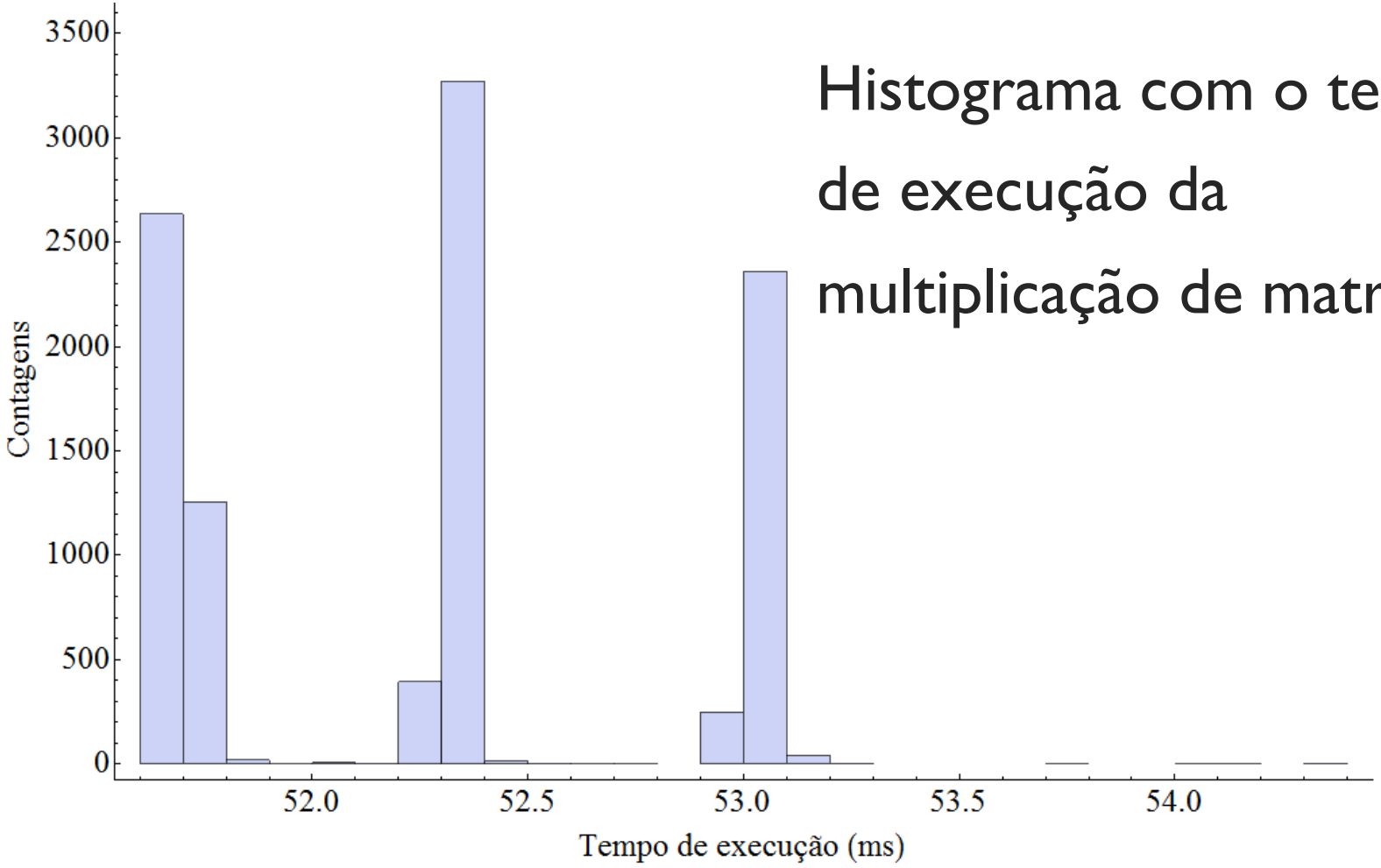
Difíceis de se detectar e não podem ser quantificados estatisticamente.

Exemplo: comparação de desempenho de GPUs.

Controle de Erros Experimentais

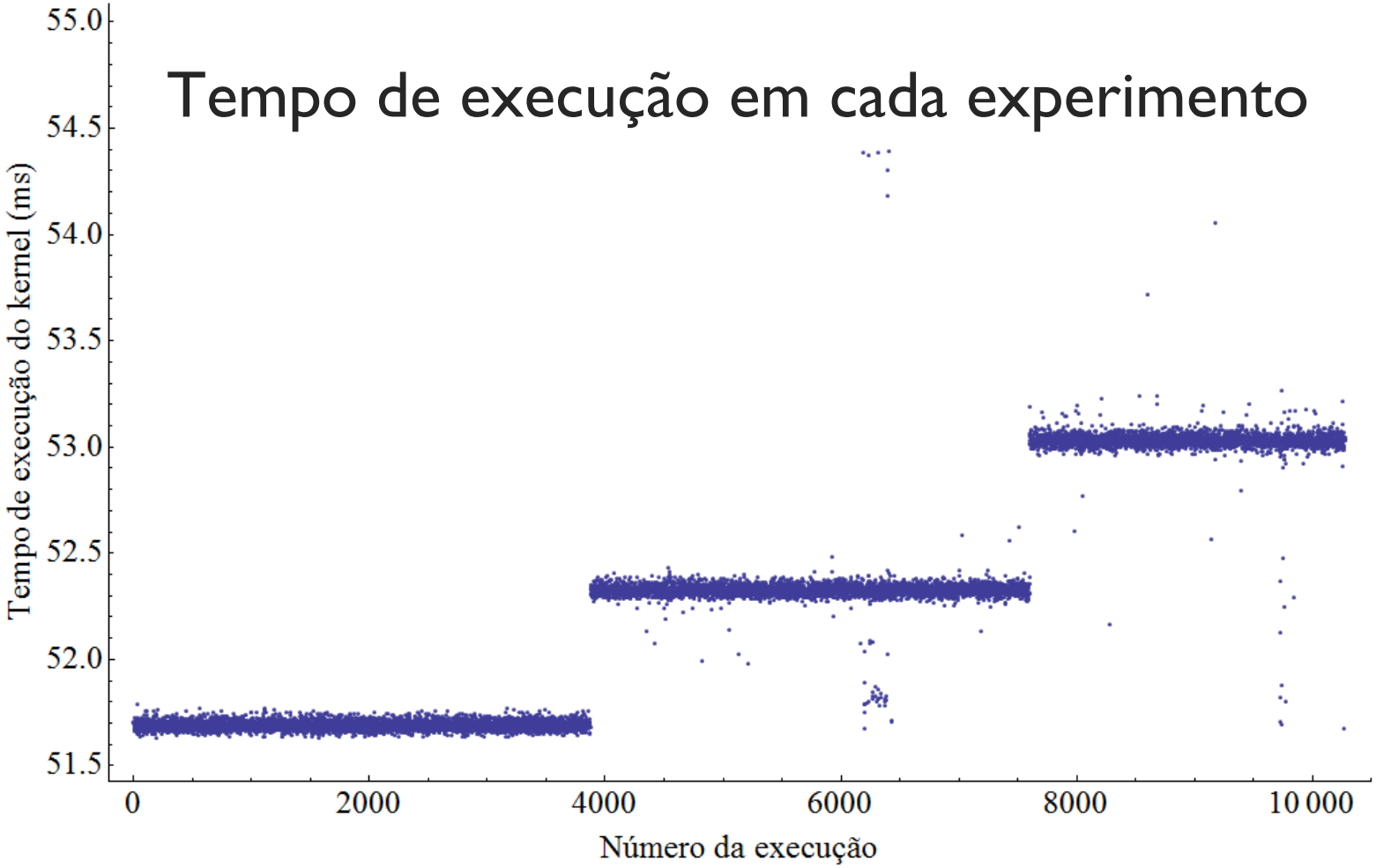


Controle de Erros Experimentais

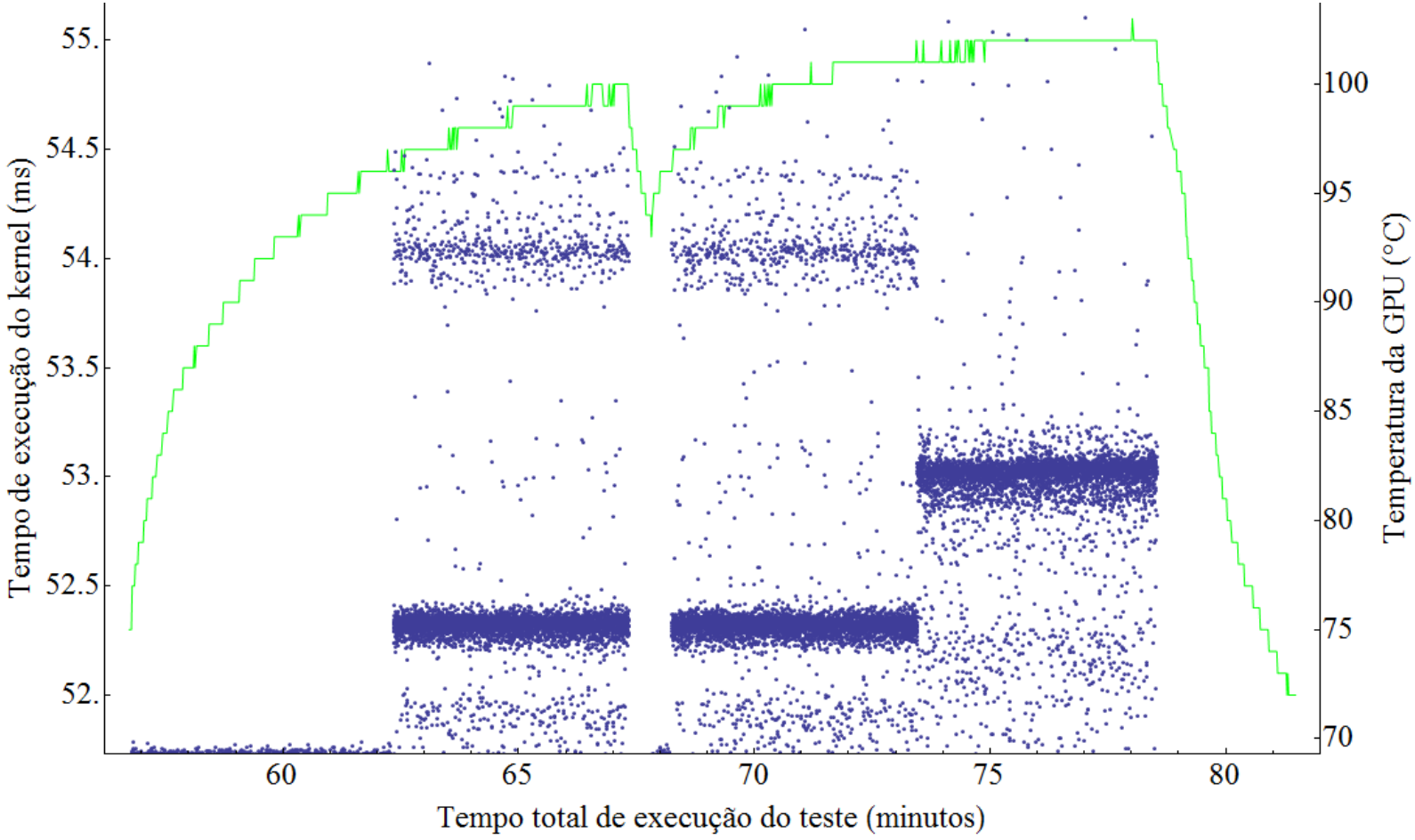


Histograma com o tempo de execução da multiplicação de matrizes

Controle de Erros Experimentais



Controle de Erros Experimentais



Metodologia de Experimentação

Resumo:

- Garantir reprodutibilidade
- Medir erros aleatórios

Atividade de laboratório

www.ic.unicamp.br/~edson/disciplinas/lncc14/index.html

- Perfilamento de Código