

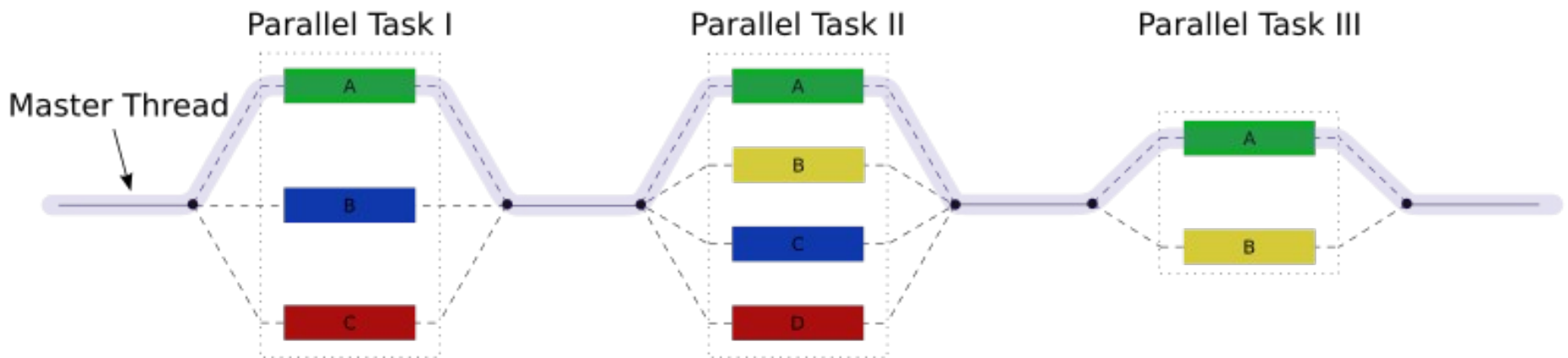
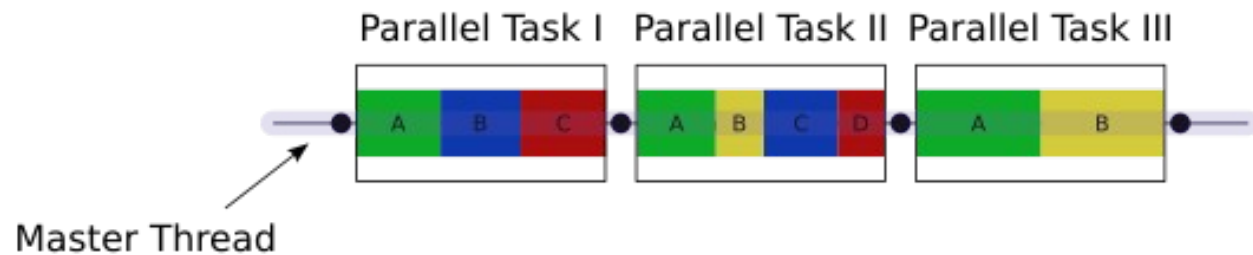
SuperComputação

Aula 9 – Ferramentas e relatórios em SuperComputação

2018 – Engenharia

Igor Montagner, Luciano Soares [<igorsm1@insper.edu.br>](mailto:igorsm1@insper.edu.br)

Aulas passadas



Aulas passadas

- 1) Más práticas de programação dificultam paralelização;
- 2) Alguns problemas são inerentemente sequenciais;
- 3) *Thread-safe*

Hoje

- 1) Revisão de thread safety + race conditions
- 2) Feedback dos relatórios SIMD
- 3) Entregas e sequência do curso

Thread safety

*garantir que não existe interação não
intencional entre threads*

- 1) Evitar estado compartilhado/efeitos colaterais
- 2) Utilizar primitivas de sincronização para acessar os dados compartilhados

Thread safety

*garantir que não existe interação não
intencional entre threads*

1) Evitar estado com
colaterais

Tarefas ingênuamente paralelas

2) Utilizar primitivas de sincronização para
acessar os dados compartilhados

Thread safety

*garantir que não existe interação não
intencional entre threads*

- 1) Evitar estado compartilhado/efeitos colaterais
- 2) Utilizar primitivas de sincronização para acessar os dados

Tarefas concorrentes ou sincronizadas

Race condition

saída do programa depende da ordem de execução das threads

- Acessos concorrentes a um recurso, com pelo menos uma escrita
- Temos estes problemas no *mandel.c* e no *pi_mc.c*

Race condition

```
long globalv = 1;

void alternate() {
    if (globalv > 1) {
        globalv -= 1;
    } else {
        globalv += 1;
    }
}
```

Quais valores globalv pode assumir neste programa?

Race condition

```
long globalv = 1;

void alternate() {
    if (globalv > 1) {
        globalv -= 1;
    } else {
        globalv += 1;
    }
}
```

E se rodarmos em duas threads?

Exercícios para entrega

- Partes 1 e 2 do roteiro 07 – *Thread safety*
- **Entrega:** 12/09

Comentários sobre relatório SIMD

Aspectos negativos

- Automação de compilação/execução
- Atenção com flags de compilação

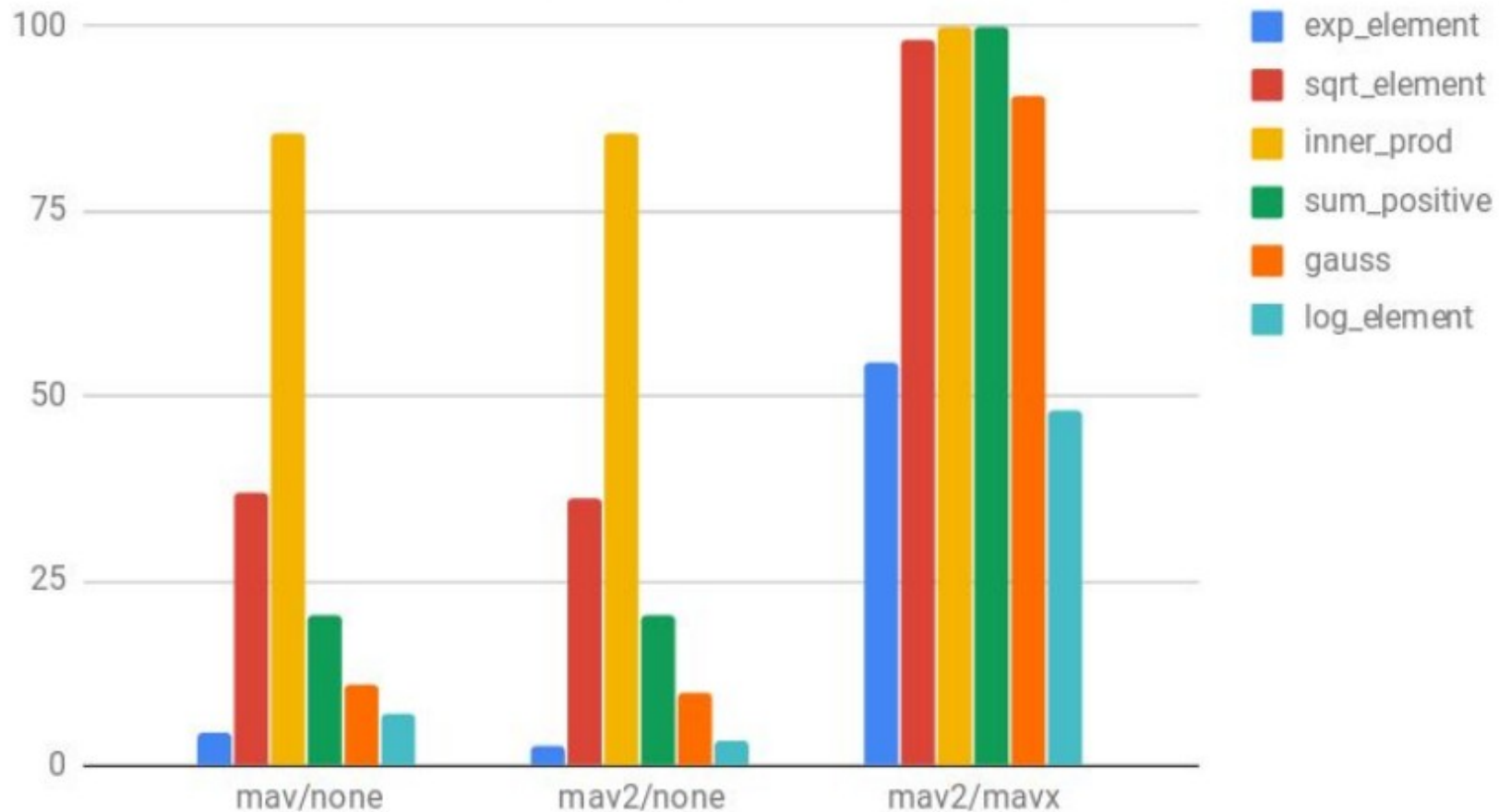
Comentários sobre relatório SIMD

Aspectos positivos

- Desempenho esperado **x** obtido
- Automatização (parcial) da geração dos gráficos

Comentários sobre relatório SIMD

Ganhos de tempo em por tipo de compilação (%)



Comentários sobre relatório SIMD

jupyter Relatório sobre desempenho SIMD_gabrielacfa_tentativa_2018-08-13-13-05-59_Tarefa3 - Avaliação de desempenho (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted

Python 3

Save Add Open Recent Up Down Run Restart Undo Redo Markdown

TAREFA 3 – AVALIAÇÃO DE DESEMPENHO


Gabriela Almeida

É possível tornar um programa mais eficiente usando opções do compilador para gerar instruções SIMD. O objetivo dessa tarefa é quantificar essa diferença de desempenho usando funções que consistem em manipulações matemáticas em vetores de tamanhos variados.

Essas diferentes funções estão presentes no arquivo *funcs.cpp*, o qual ao ser compilado e executado gera 21 vetores com conteúdos e tamanhos variados (de 100 a 100 milhões de elementos) e calcula o tempo que demorar para executar cada função em todos os elementos desses vetores. Por fim é gerado um arquivo *t3Tempos.txt* que contem um vetor chamado "tamanhos" o qual possui os tamanhos dos vetores usados para fazer os cálculos em cada iteração. Além disso, contem vetores para cada tipo de função, o qual possui o tempo gasto para fazer aquele cálculo naquela iteração.


```
In [1]: from IPython.display import Image
        Image("Figura1.png")
```

Ferramentas - Relatório

 Pweave documentation

0.30

Search docs

 Pweave - Scientific Reports Using Python

Features:

Install and quickstart:

Documentation

Thanks

Pweave Basics

Using Pweave from Command Line

Code Chunk Options

Output Formats

Using pweave module

Using Bokeh with Pweave

Changing defaults

Publishing scripts

Customizing output

Subclassing formatters

Editor support

[Docs](#) » Pweave - Scientific Reports Using Python

[View page source](#)

Pweave - Scientific Reports Using Python

Pweave is a scientific report generator and a literate programming tool for Python. Pweave can capture the results and plots from data analysis and works well with NumPy, SciPy and matplotlib. It is able to run python code from source document and include the results and capture matplotlib plots in the output.

Note

Pweave 0.30 has been updated to use IPython to run code from the documents. This brings support for IPython magics and rich output and support for other.

Features:

- Python 3.4, 3.5 and 3.6 compatibility
- **Execute python code** in the chunks and **capture** input and output to a report.
- Rich output and support for IPython magics
- **Use hidden code chunks**, i.e. code is executed, but not printed in the output file.
- Capture matplotlib graphics.
- Evaluate inline code in documentation chunks marked using `<% %>` and `<%= %>`.
- Cache all code and results from previous runs for fast report generation when you are only working with documentation. Inline code will be hidden in documentation mode.

PWeave

Ferramentas

Aprendam a usar um Debugger!

- VSCode
- KDevelop
- GDB linha de comando [1] [2]

Referências

- Livros:
 - Hager, G. ; Wellein, G. **Introduction to High Performance Computing for Scientists and Engineers**. 1ª Ed. CRC Press, 2010.
- Artigos:
 - Dagum, Leonardo, and Ramesh Menon. "OpenMP: an industry standard API for shared-memory programming." *IEEE computational science and engineering* 5, no. 1 (1998): 46-55.
- Internet:
 - <https://www.youtube.com/playlist?list=PLLX-Q6B8xqZ8n8bwjGdzBJ25X2utwnoEG>
 - <http://www.openmp.org/wp-content/uploads/omp-hands-on-SC08.pdf>
 - http://extremecomputingtraining.anl.gov/files/2016/08/Mattson_830a_ug3_HandsOnIntro.pdf

Insper

www.insper.edu.br