Threads e Processos

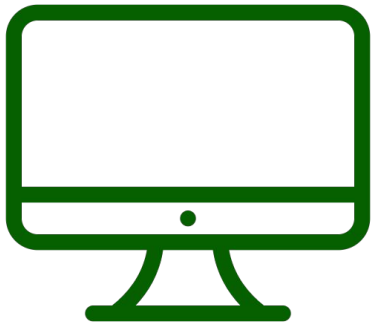
Implementação e mapeamento do desempenho

Alunos:
Leonardo Araujo Armelin
Cleiton da Silva Guilhermite
Stefany Figueiredo
Luiz Felipe Miranda

Professor: Marco Aurélio Ferreira

Processos

- Programa em execução
- Espaço de endereçamento próprio
- Recursos alocados
- Contexto de execução



Threads

- Menor unidade de execução
- Threads múltiplas
- Compartilhamento de recursos
- Threads de usuário
- Threads de núcleo

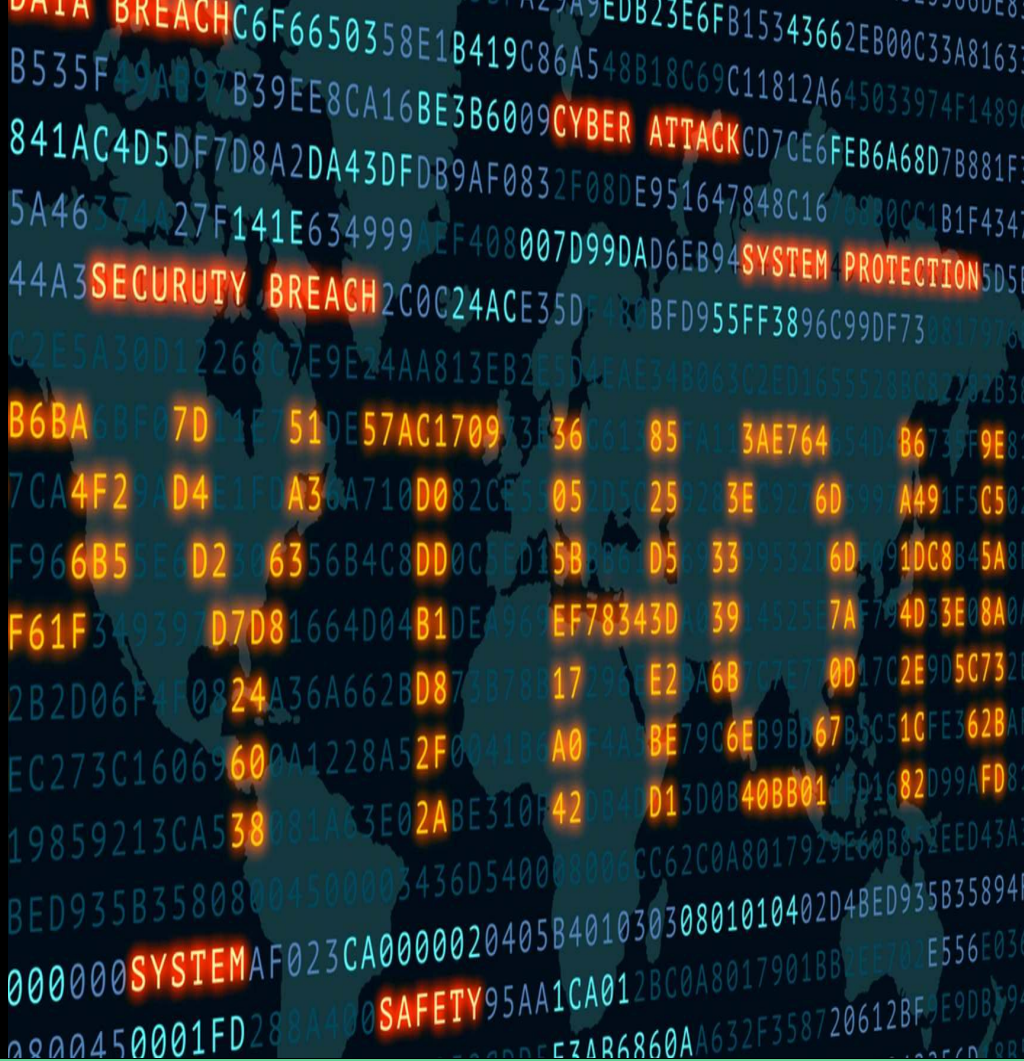
Corolário

Aplicação estruturada como processo para processamento CPU-Bound e operação E/S não oferece vantagens de desempenho em relação a aplicações estruturada como threads para esse tipo de processamento; Diferentemente que se estruturada como thread.

Objetivos

Implementar um algoritmo que divida um comprimento por um diâmetro de uma circunferência iterativamente de forma que comprove a afirmação sobre desempenho de processos e threads do corolário e determine o ponto de inflexão através de uma análise com notação assintótica.

Implementação do algoritmo



Página GitHub

Para uma visualização ampla dos conteúdos presentes no trabalho, assim como os resultados obtidos, acesse nossa página do GitHub por meio do link abaixo:

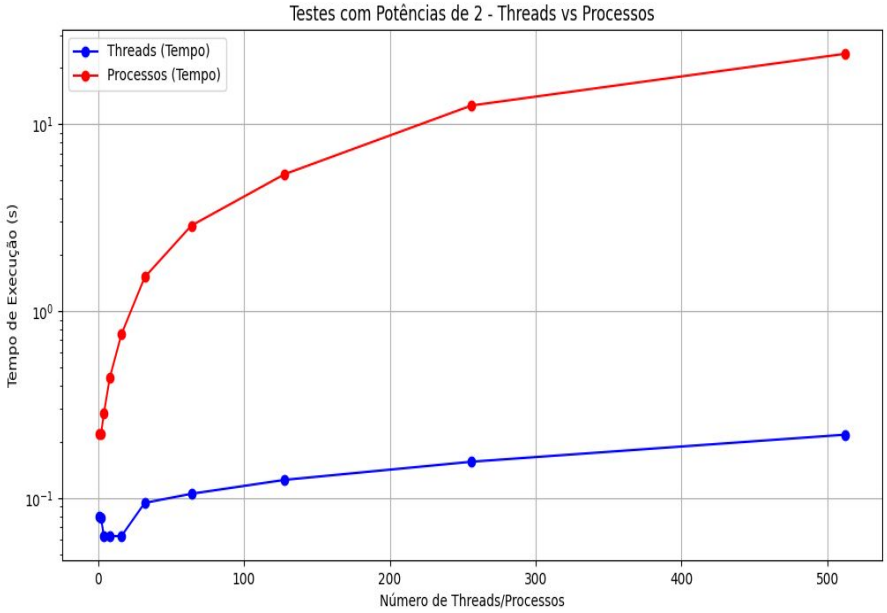
Click here!

DISCUSSÃO DOS RESULTADOS

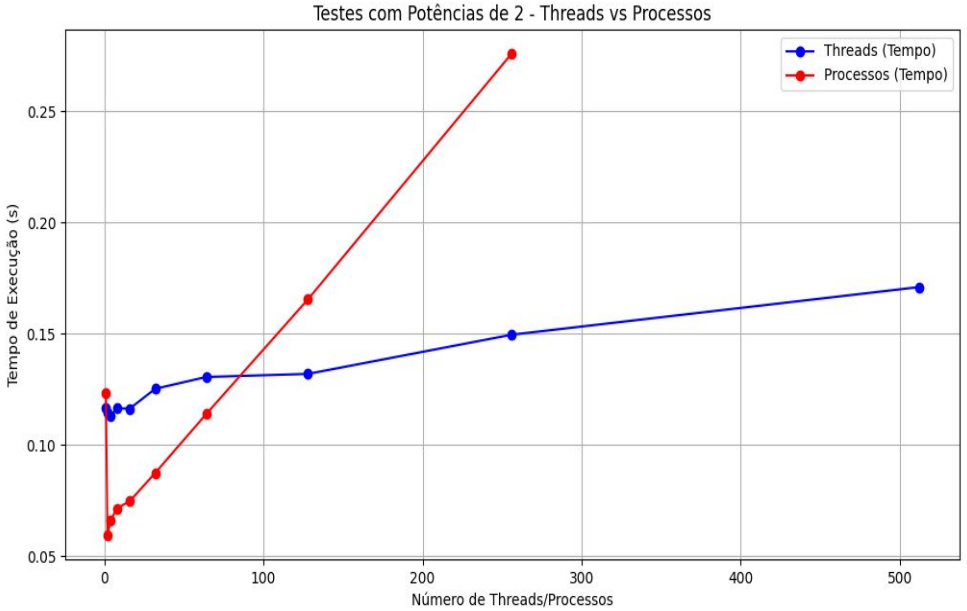
A comparação do desempenho com as iteração implementadas por meio de Threads e Processos possibilitaram em uma melhor compreensão das vantagens de cada aplicação.

Análise - Testes com potências de 2

Dispositivo 1



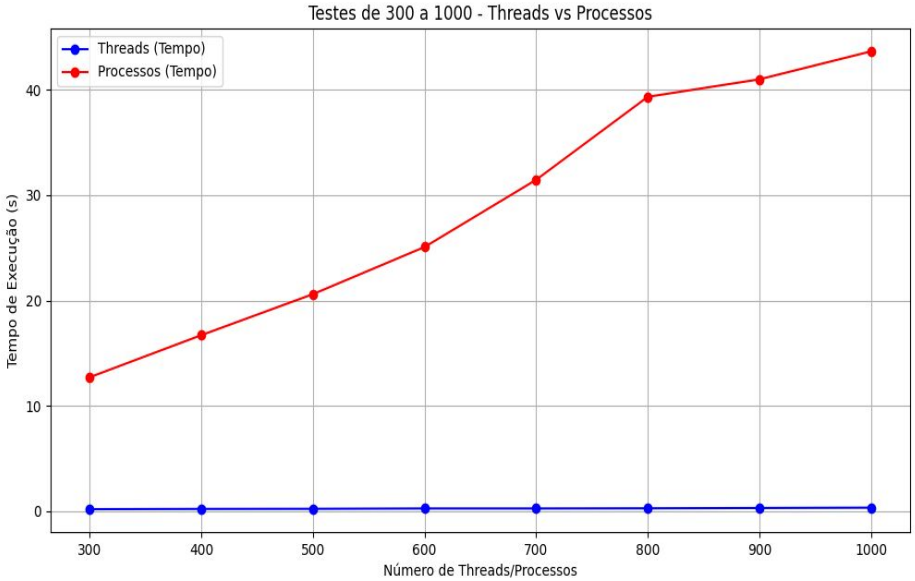
Dispositivo 2



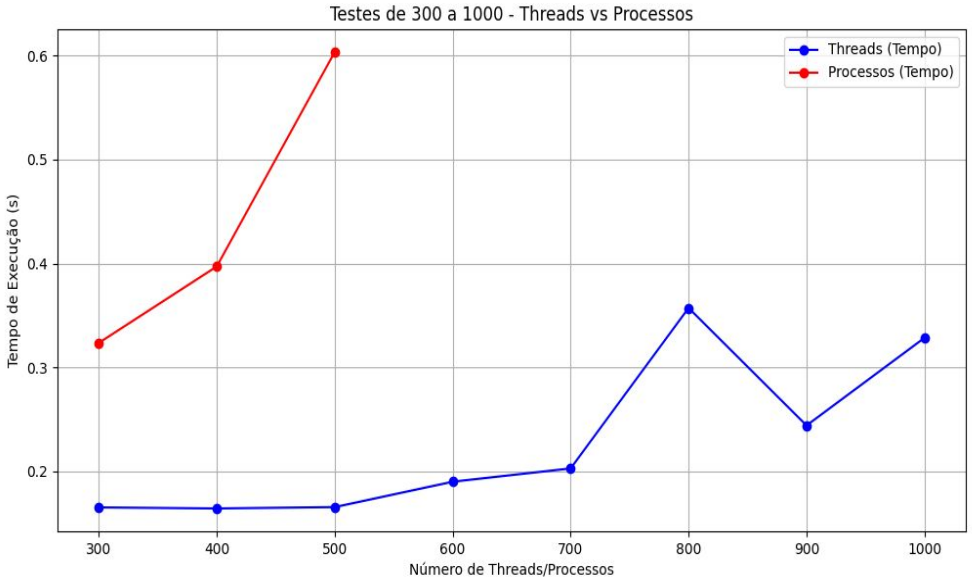
Obs.: Tempos de threads com crescimento gradual e a ineficiência dos processos com seu aumento considerável. Este único teste para o dispositivo2 apresentou interseção nos tempos de operação

Análise - Testes de 300 a 100

Dispositivo 1



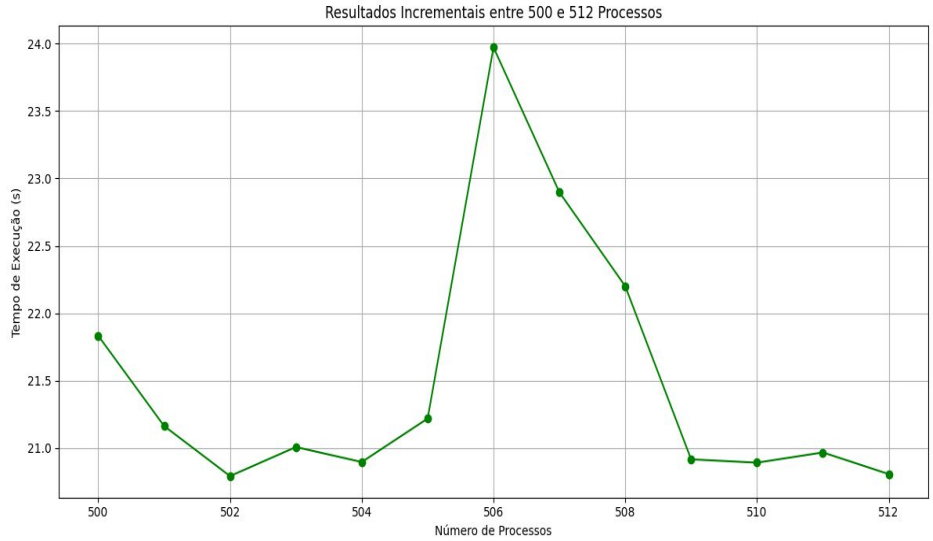
Dispositivo 2



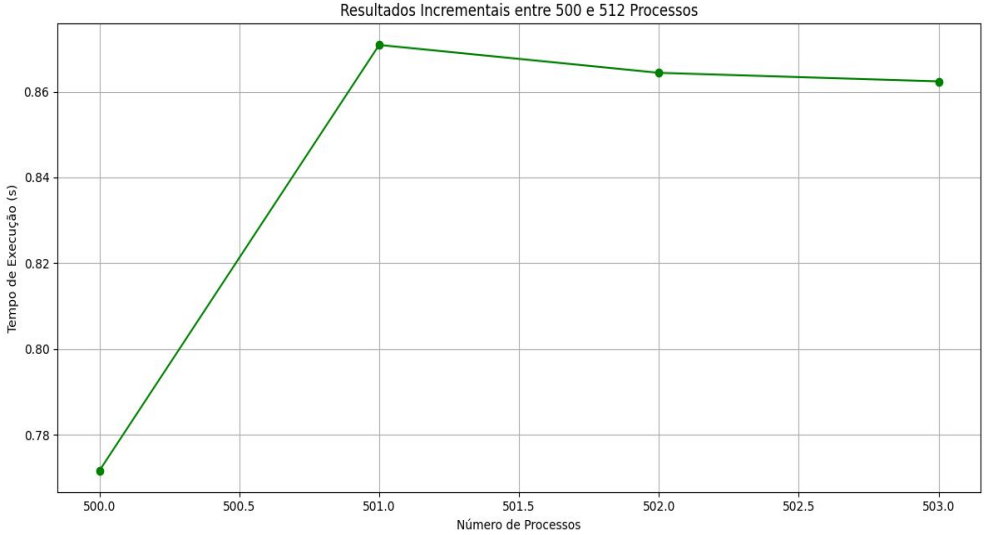
Obs.: Dispositivo1 com melhor visualização e comprovação do corolário quanto a eficiência de na realização de operações de E/S com um processamento estruturado em threads.

Análise - Testes de 500 a 512 processos

Dispositivo 1



Dispositivo 2



Obs.: Em ambos os testes, os tempos para processamento dos dos processos estão irregulares. Para o dispositivo 2, o limite crítico já foi atingido.

CONSIDERAÇÕES FINAIS

- Validação do corolário apresentado.
- Ganho de desempenho Threads X Processos.
- O uso do GIL.
- Uma abordagem para os deadlocks em ambientes de concorrência;
- Possíveis riscos inerentes à concorrência.

Validação do corolário e ganho de desempenho

Processos estruturados não têm vantagens significativas em operações CPU-Bound.

- Threads: Desempenho estável, influenciado pelo GIL e compartilhamento de memória.
- Processos: Desempenho inicial competitivo, mas custo de criação/gerenciamento aumenta exponencialmente com mais processos.
- Recomendação: Em CPU-Bound, threads são mais eficientes sob restrições. Em contrapartida, processos são vantajosos apenas em pequenas quantidades.

O uso do GIL (Global Interpreter Lock)

- Impacto do GIL: Simplifica a implementação e evita condições de corrida, mas limita a escalabilidade de threads em tarefas CPU-Bound.
- Recomendação: Para cálculos intensivos, prefira multiprocessing em Python.
- Conclusão: Resultados destacam o impacto do GIL, o overhead de processos e a eficácia de abordagens de paralelismo, alinhados a conceitos de context-switching, paralelismo verdadeiro e concorrência.

Os deadlocks em ambientes de concorrência

- Desafios em concorrência: Possibilidade de deadlocks e condições de corrida em acesso simultâneo a recursos compartilhados.
- Observação: Embora o código analisado esteja livre de deadlocks, cenários complexos, como filas compartilhadas, podem apresentar riscos se travas forem mal gerenciadas.
- Resultado: Na implementação atual, o gerenciamento de travas foi adequado, evitando problemas de concorrência.

Possíveis riscos

Threads e processos operam de forma independente, sem travas compartilhadas, dependências ou comunicação entre estruturas. Sendo assim, riscos são descartados na implementação do algoritmo, devido a sua ausência por já serem tratados.

- Condições de corrida:
- Vantagem: Threads compartilham memória, eficiente para tarefas independentes.
- Desafio: Manipulação concorrente de variáveis globais permanece arriscada.
- Impacto do GIL: Reduz probabilidades de condições de corrida, mas não elimina problemas em sistemas que exigem sincronização explícita.

Obrigado!
Em agradecimento a todos os contribuintes

Referências

https://logos-world.net/wp-content/uploads/2020/11/GitHub-Symbol.png

https://th.bing.com/th/id/R.39586efcf32a93aa68847d7a46345874?rik=lxU%2fI8ZMUBGvYg&pid=ImgRaw&r=0

https://cadernointeligente.pt/storage/images/image?remote=https:%2F%2Fcadernointeligente.pt%2FWebRoot%2FClaranet%2FShops%2Floja101263%2F5E3E%2FCB6B%2FE5EC%2FF234%2F2505%2F0A0C%2F05B9%2F00BC%2Fverde-pastel-grande-01.jpg&shop=loja101263&width=600&height=2560

TANENBAUM, A. S. Sistemas operacionais modernos / Andrew S. Tanenbaum; tradução Ronaldo A.L. Gonçalves, Luís A. Consularo, Luciana do Amaral Teixeira; revisão técnica Raphael Y. de Camargo. – 3. Ed – São Paulo: Prentice-Hall Do Brasil, 2010. 653 p. ISBN 9788576052371.

JORDÃO, Fabio. O que são threads em um processador?. Tecmundo, 18 abr. 2011. Disponível em: https://www.tecmundo.com.br/9669-o-que-sao-threads-em-um-processador-.htm. Acesso em: 18 dez. 2024.

BEAZLEY, David. Understanding the Python GIL. PyCon 2010, Atlanta, GA, 2010. Disponível em: https://www.youtube.com/watch?v=Obt-vMVdM8s. Acesso em: 18 dez. 2024.

TANENBAUM, A. S.; BOS, H. Modern Operating Systems. 4ª ed. Pearson, 2014.

STUART, Brian L. Princípios de sistemas operacionais: projetos e aplicações. São Paulo: Cengage Learning, 2011. ISBN 9788522107339.