

Análise de desempenho de processos

Cleomacio Silva de Oliveira RA: 817122831
Diego Ramires Fernandes RA: 820120772
Lucas de Freitas Pinheiro RA: 81727525
Murilo Yudji Hara RA: 81726559
Vinícius Oliveira Gonçalves Silva RA: 81723868

Resumo: O estudo visa realizar análise de processos em diferentes conceitos de programação abordando metodologias usadas com as devidas comparações entre os resultados entre os projetos apresentados.

1 Introdução

A evolução proveniente da informática no cenário atual é muito rápida e de tempos em tempos é lançado uma nova tecnologia trazendo consigo benefícios no quesito espaço/tamanho, performance e tempo. É evidente que sempre que há um novo lançamento de periféricos como por exemplo, haja uma reestruturação total no restante dos componentes, afinal é preciso haver o sincronismo entre todas as partes envolvidas neste processo. Neste quesito, em contrapartida que há a evolução na parte física (hardware), também há a necessidade de dar cumprimentos e boas-vindas a evolução a parte lógica (software).

Novas linguagens de programação também são anunciadas e trazem consigo aspectos modernos que acompanham o ritmo de desenvolvimento. Com a chegada de processadores de multiprocessamento como a “família” core por exemplo, é evidente a utilização de alguns mecanismos como paralelismo, delegando a distribuição de funções do código fonte para cada núcleo do processador, afim de aumentar o poder de processamento, aumentando a produtividade e diminuindo o tempo de execução de funções, métodos ou tarefas.

O objetivo do projeto consiste basicamente em realizar o processamento de determinada tarefa envolvendo neste caso, cálculos aritméticos onde será executado em diferentes conceitos de programação, obtendo como resultado final o tempo de execução desta tarefa para análise de performance e desempenho.

2 Detalhamento conceitual

Serão utilizados três projetos que oferecem recursos diferentes:

- Projeto com a estrutura sequencial;
- Projeto com a utilização de Threads;
- Projeto com a utilização de multiprocessamento;

Cada projeto possui sua estrutura e uma função cujo objetivo é verificar se o número dentro de determinado range é um número primo para o cálculo de desempenho. Os valores de range serão obviamente iguais para todos os projetos, para que seja possível termos uma base concreta de resultados.

A função recebe 3 parâmetros contendo ID, valor inicial(range) e valor final (range) e percorre o range identificando os números primos dentro deste intervalo.

Para cada chamada à função, será contabilizado:

1. Tempo total de execução da função;
2. Quantidade total de números primos encontrados dentro do range especificado;
3. “Lista” ou vetor que contém os números primos encontrados;

Esses valores serão armazenados em uma tabela para ao final de todas execuções de todos os projetos podermos chegar a uma conclusão final em relação ao desempenho de cada projeto.

3 Implementação dos projetos

3.1 Projeto com a estrutura sequencial

Este projeto possui a estrutura sequencial comumente utilizada pela maioria de programas espalhados na net, onde a linha subsequente só será executada após o término da linha corrente.

A função “gerar_primos” é chamada uma única vez gerando os resultados.

3.2 Projeto com a utilização de Threads

Este projeto faz a utilização de subprocessos compartilhando a mesma região de memória (espécie de processos filhos proveniente de um processo pai). Para cada subprocesso criado, será distribuído um valor de range para verificação de números primos onde estará realizando os devidos procedimentos. Normalmente espera-se uma melhor produtividade em relação ao tempo com esta estrutura de projeto, mas isto não acontece em razão da abordagem da linguagem com Threads (ver em detalhes da implementação).

Em cada execução deste projeto, a função “gerar_primos” é chamada oito vezes, contabilizando oito subprocessos.

Geralmente, este tipo de estrutura de projeto é melhor utilizado quando há um fluxo de dados trabalhando com E/S (entrada e saída), em determinado tempo onde o tempo de espera sempre é alto, mantendo o sistema responsável durante este para sua utilização.

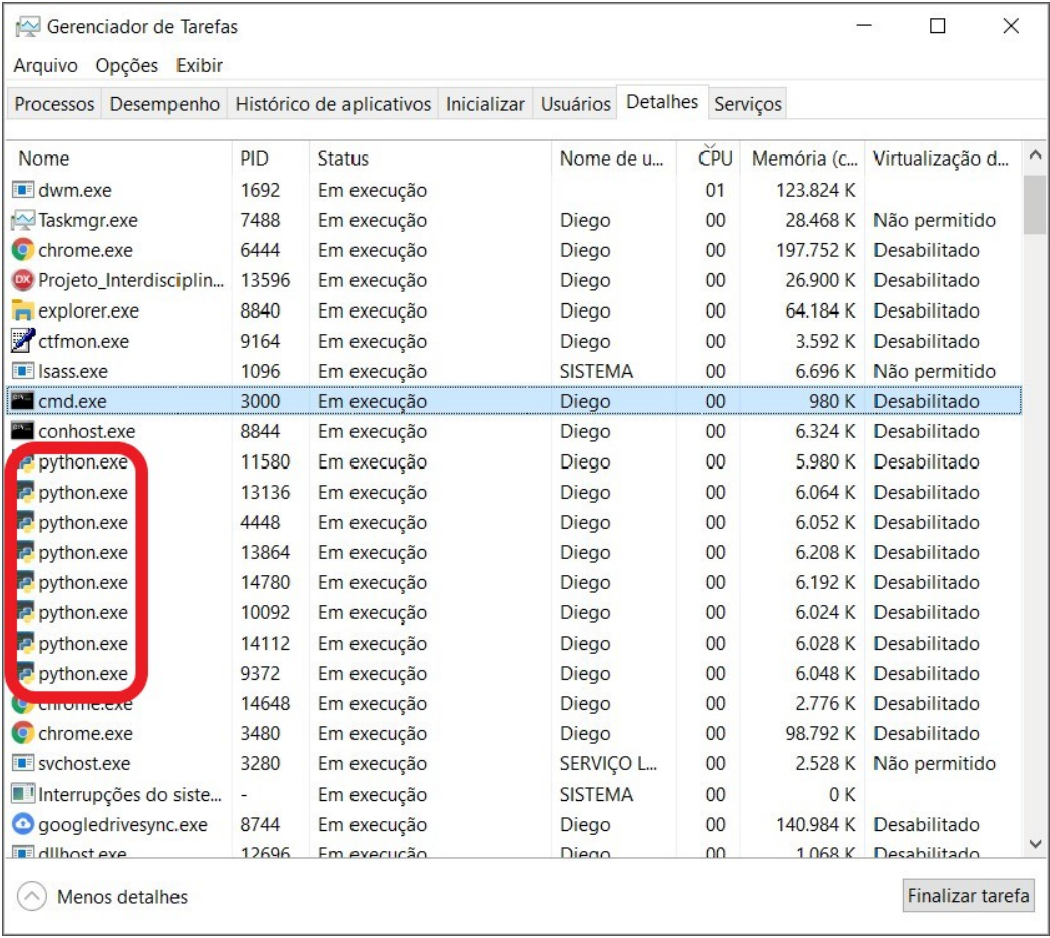
Exemplo de utilização em projetos:

- Consulta a banco de dados
- Monitoramento de tráfego em projeto de redes
- Games

3.3 Projeto com a utilização de multiprocessamento

Este projeto faz a utilização de processos independentes ao invés de subprocessos, isto é, não faz o uso de compartilhamento da mesma região de memória, fazendo o uso de outros núcleos de processamento disponíveis para trabalhar em favor da função.

A títulos didáticos, é interessante abrir o gerenciador de tarefas (CTRL + ALT + DEL) na aba “Detalhes” e verificar a criação dos processos durante esta execução do projeto, assim pode-se ver os processos sendo criados e a medida que eles vão finalizando cada range, eles são finalizados automaticamente.



Nome	PID	Status	Nome de u...	CPU	Memória (c...	Virtualização d...
dwm.exe	1692	Em execução		01	123.824 K	
Taskmgr.exe	7488	Em execução	Diego	00	28.468 K	Não permitido
chrome.exe	6444	Em execução	Diego	00	197.752 K	Desabilitado
Projeto_Interdisciplin...	13596	Em execução	Diego	00	26.900 K	Desabilitado
explorer.exe	8840	Em execução	Diego	00	64.184 K	Desabilitado
ctfmon.exe	9164	Em execução	Diego	00	3.592 K	Desabilitado
lsass.exe	1096	Em execução	SISTEMA	00	6.696 K	Não permitido
cmd.exe	3000	Em execução	Diego	00	980 K	Desabilitado
conhost.exe	8844	Em execução	Diego	00	6.324 K	Desabilitado
python.exe	11580	Em execução	Diego	00	5.980 K	Desabilitado
python.exe	13136	Em execução	Diego	00	6.064 K	Desabilitado
python.exe	4448	Em execução	Diego	00	6.052 K	Desabilitado
python.exe	13864	Em execução	Diego	00	6.208 K	Desabilitado
python.exe	14780	Em execução	Diego	00	6.192 K	Desabilitado
python.exe	10092	Em execução	Diego	00	6.024 K	Desabilitado
python.exe	14112	Em execução	Diego	00	6.028 K	Desabilitado
python.exe	9372	Em execução	Diego	00	6.048 K	Desabilitado
chrome.exe	14648	Em execução	Diego	00	2.776 K	Desabilitado
chrome.exe	3480	Em execução	Diego	00	98.792 K	Desabilitado
svchost.exe	3280	Em execução	SERVIÇO L...	00	2.528 K	Não permitido
Interrupções do siste...	-	Em execução	SISTEMA	00	0 K	
googledrivesync.exe	8744	Em execução	Diego	00	140.984 K	Desabilitado
dllhost.exe	12696	Em execução	Diego	00	1.068 K	Desabilitado

FIGURA 1

FONTE: GERENCIADOR DE TAREFAS (DETALHES)

Sua implementação é semelhante as threads, porém obviamente com suas reservadas particularidades.

Em cada execução deste projeto, a função “gerar_primos” é chamada oito vezes, contabilizando oito processos.

É sempre importante levar em consideração a utilização desta estrutura de projeto quando a finalidade proposta exige uso de CPU (processamento) para alcançar a melhor eficiência e produtividade destes processos a fim de obter melhor desempenho e performance. Alguns exemplos de utilizações que podem ser utilizados:

- Cálculos aritméticos
- Processamento e reconhecimento de imagem
- Algoritmos de busca e ordenação em memória

4 Detalhes da implementação

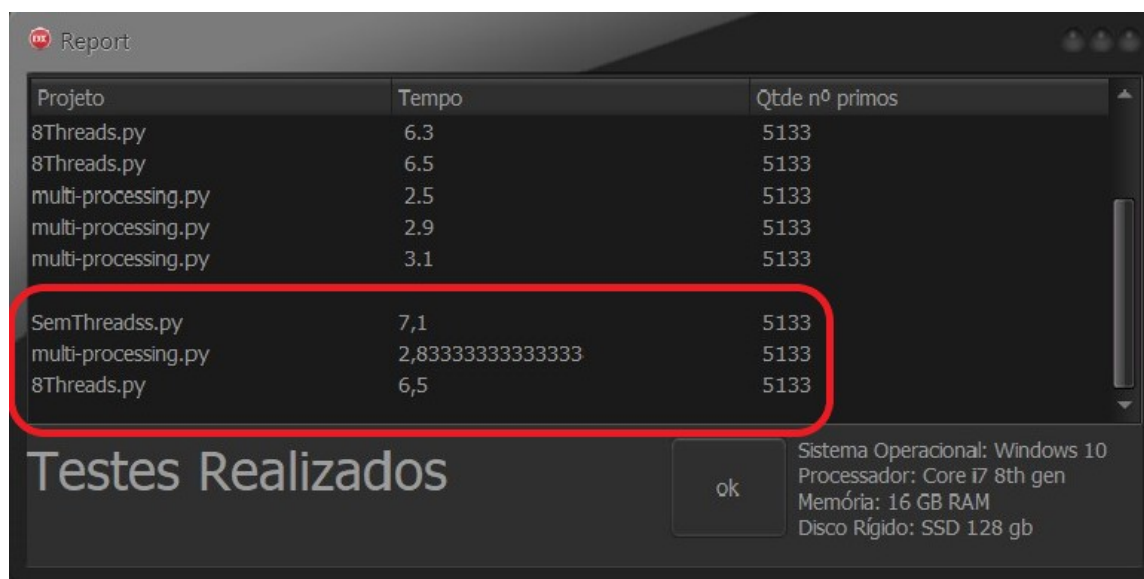
A linguagem de programação que foi utilizada para a execução deste projeto é Python. Por ser uma linguagem de alto nível, facilita sua utilização levando algumas características do seu uso como orientação a objetos, sintaxe simples, disponíveis para as mais diversas plataformas, transparente e muito elegante porém é interessante observar um comportamento nesta linguagem que é interpretada, quando se faz a utilização de Threads distribuindo a função de acordo com cada range especificado no código fonte, onde o interpretador desta linguagem utiliza um bloqueio em memória (GIL - Global Interpreter Lock) que impede que outros núcleos possam ser acionados para auxiliar na tarefa para que seja melhorado o desempenho da função/ tarefa. Em razão desta ineficiência, foi proposto e elaborado um terceiro projeto contendo a estrutura de multiprocessamento, que aborda uma semelhança aos conceitos de threads (foi preciso a alteração em até 3 linhas no máximo em cima do projeto referido) porém com a utilização de processos independentes, “quebrando” esta barreira de proteção em memória que faz com que impeça de acionar os núcleos de processamento, otimizando a performance e produtividade do nosso projeto através destes. Cada projeto foi executado 3 vezes para obtermos uma média concreta no final dos testes, através de uma média aritmética.

Os valores apresentados através das considerações finais podem apresentar variações se executados em determinados tempos em outras ocasiões especiais já que os núcleos sofrem concorrência em determinadas situações.

Os testes foram realizados em um sistema operacional Windows 10, processador Intel Core i7 8th Gen, 16 GB RAM, SSD 128 GB.

5 Considerações Finais

Entre os três projetos que foram executados conforme a tabela abaixo, representado pelo nome do projeto, tempo em milissegundos e a quantidade de números primos que foram encontradas dentro do range do número 0 a 50.000, pode-se observar:



Projeto	Tempo	Qtde nº primos
8Threads.py	6.3	5133
8Threads.py	6.5	5133
multi-processing.py	2.5	5133
multi-processing.py	2.9	5133
multi-processing.py	3.1	5133
SemThreadss.py	7,1	5133
multi-processing.py	2,8333333333333333	5133
8Threads.py	6,5	5133

Testes Realizados

ok

Sistema Operacional: Windows 10
Processador: Core i7 8th gen
Memória: 16 GB RAM
Disco Rígido: SSD 128 gb

FIGURA 2

FONTE: TABELA DE RESULTADOS (REPORT)

O projeto que fez a utilização de multiprocessamento utilizando outros núcleos de processamento foi o que obteve a melhor performance e desempenho em tempo, onde teve aproximadamente uma redução de aproximadamente 60% em comparação ao projeto de uso sequencial.

Também é notável a redução insignificante de performance e desempenho em tempo em relação ao projeto sequencial com o projeto que faz a utilização de threads, em razão da peculiaridade adotada na linguagem em relação a proteção de memória (ver em detalhes da implementação) e a complexidade da função.

A delegação do poder de processamento de acordo com a utilização da CPU pode ser observada na figura abaixo:

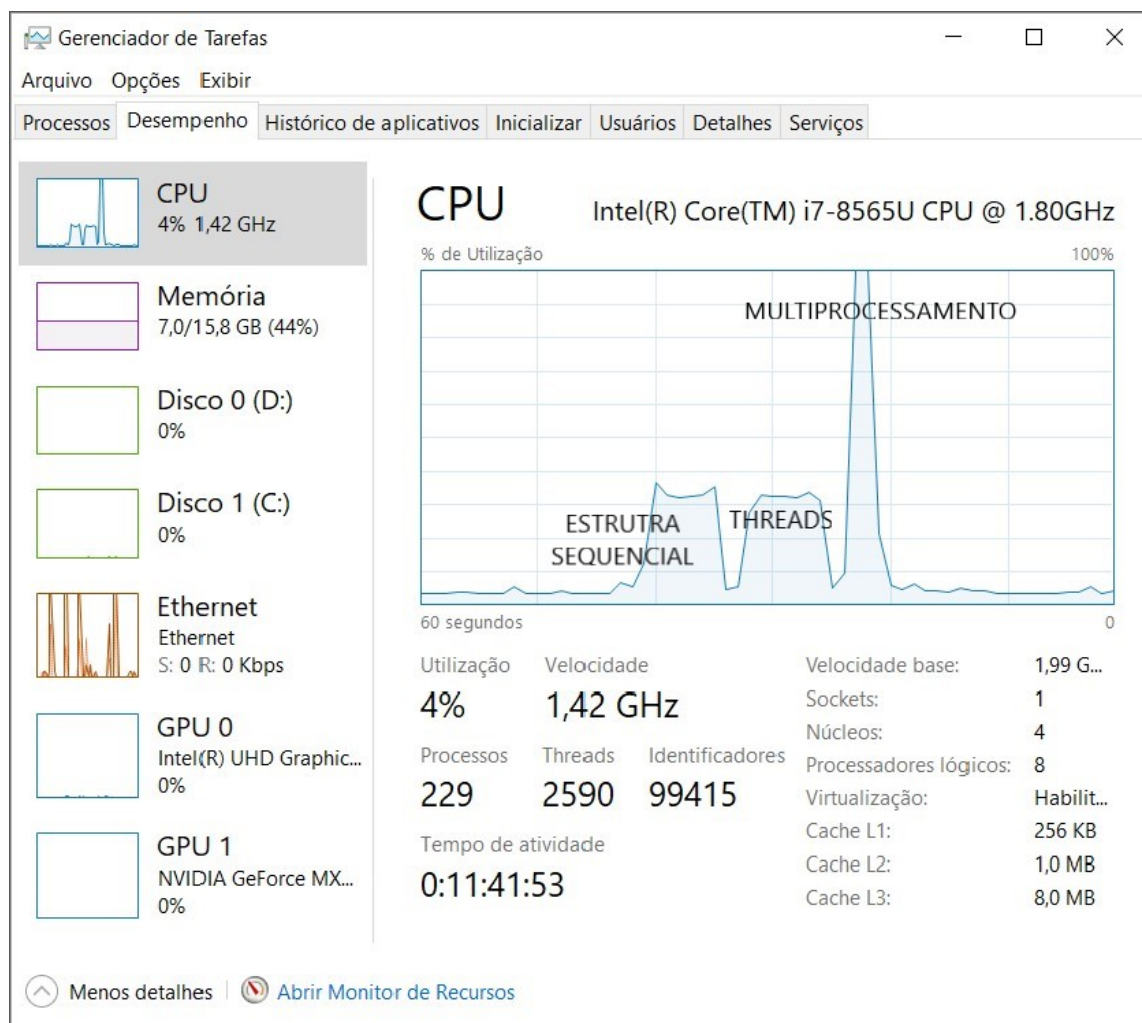


FIGURA 3

FONTE: GERENCIADOR DE TAREFAS (DESEMPENHO)

Apêndice

Código fonte: <https://github.com/rfdiego/AnaliseEDesempenhoDeProcessos>

Referências

- [1] BEAZLEY, David. "Python Cookbook: Receitas para dominar Python 3." São Paulo: Novatec, 2013. 720 p.
- [2] K. PRASAD, Sushil. L.ROSENBERG, Arnold. SUSSMAN, Alan. C. WEEMS, Charles. GUPTA, Anshul "Topics in Parallel and Distributed Computing: Introducing Concurrency in Undergraduate Courses" Waltham: Morgan Kaufmann, 2015. 360 p.

[3] P. LANGTANGEN, Hans. "A Primer on Scientific Programming with Python: 6" London: Springer, 2016. 922 p.

[4] Python.org. Python 3.8.3: The Python Standard Library, 2020. Documentation. Disponível em: <https://docs.python.org/3/library/index.html>. Acesso em: 05 de jun. de 2020.