12 FEPEG FÓRUM ENSINO · PESQUISA · EXTENSÃO · GESTÃO
CIÊNCIA E TECNOLOGIA: IMPLICAÇÕES NO ENSINO, PESQUISA E EXTENSÃO
REALIZAÇÃO: Unimontes PIBID Unimontes
APOIO: FAPEMIG FADENOR
ISSN: 1806-549X

# USO DE PARALELISMO NA LINGUAGEM DE PROGRAMAÇÃO PYTHON

**Autores:** ALLYSSON COSTA E SILVA, MAURILIO JOSÉ INÁCIO

## Introdução

A evolução dos processadores modernos tem permitido ganhos expressivos quanto à paralelização de tarefas que podem ser executadas concorrentemente em sistemas computacionais. Duas vias principais podem ser encontradas atualmente no mercado: sistemas com múltiplos processadores e sistemas *multicore* (HENNESSY et al., 2014). A execução de *threads* e processos em um sistema operacional pode se beneficiar de vários elementos processadores uma vez que conseguem rodar simultaneamente no sistema computacional onde estão sendo executados. Uma *thread* é a menor unidade de execução controlada por um sistema operacional moderno (KAMAL, 2015). Este trabalho tem como objetivo realizar um estudo comparativo entre a execução serial e paralela de processos e *threads* utilizando a linguagem de programação Python em um sistema *multicore*. Neste ponto propõem-se a seguinte questão de pesquisa: No contexto da linguagem Python, dentre as modalidades de processamento serial e paralelo, qual vertente alcança menor tempo de execução para uma tarefa processamento de linguagem natural?

## Material e métodos

O fluxo de atividades a serem realizadas neste trabalho será detalhado nesta seção. Este compreende a escolha da tarefa computacional a ser executada, o modelo de instanciação, uso e finalização de *threads* e processos, além da escolha da arquitetura de hardware e software necessários para execução do experimento.

### *A. Tarefa computacional escolhida*

A tarefa escolhida para ser executada neste trabalho consiste no pré-processamento de arquivos de código fonte de softwares livres escritos na linguagem de programação Java. Esta atividade é essencial para a área de Engenharia de Software no tocante à recuperação de rastreabilidade entre artefatos de software utilizando processamento de linguagem natural e recuperação de informação (PAECH; HUBNER, 2017). O pré-processamento é uma das partes constituintes de todo um processo de recuperação de rastreabilidade e consiste no processamento de informação textual advindo de artefatos de software como código fonte, descrições de *bugs* ou funcionalidades, requisitos de software, dentre outros. A recuperação da rastreabilidade permite recuperar rastros de ligação entre diversos artefatos de software como descrições de requisitos funcionais e trechos do código fonte onde os mesmos são implementados como métodos e classes.

Durante o pré-processamento são removidas *stopwords*, números, caracteres especiais e todo e qualquer vocábulo que não agregue valor semântico aos arquivos de código fonte (KUHN et al., 2007). Esta é uma típica tarefa *cpu bound*, isto é, que faz uso intensivo do processador.

### *B. Hardware e Software necessários ao experimento*

Como objeto de estudo foram utilizados seis softwares de código aberto escritos na linguagem de programação Java: ArgoUML, BlueJ, JEdit, DrJava, JFreeChart e Lucene. Em cada software foram selecionados os arquivos de código fonte totalizando 263 MB de informação bruta a ser processada.

Para realizar o pré-processamento sobre os arquivos citados utilizou-se a linguagem de programação Python e sua distribuição de pacotes denominada Anaconda que conta com as principais bibliotecas Python para uso técnico e científico. Além disso, o ambiente de desenvolvimento integrado PyCharm foi utilizado.

No tocante ao hardware os testes foram executados em um computador com processador da marca AMD com quatro núcleos lógicos e quatro núcleos físicos, modelo A10 6800K, usando um total de 16 GB de memória RAM.

ISSN: 1806-549X

*C. Cenários de execução de Threads e Processos em Python*

A linguagem de programação Python possui um mecanismo denominado GIL (*Global Interpreter Lock*) que coordena a execução de múltiplas *threads*, estando limitado a utilizar apenas um núcleo de processador (PHILLIPS et al., 2016). Dadas tais limitações, além de apresentar a performance baseada em *threads* para a realização do pré-processamento, optou-se por utilizar a biblioteca *multiprocessing* que permite a execução simultânea de múltiplos processos que se comunicam através de IPC (*Inter Process Communication*) e podem lançar mão de vários núcleos do processador durante a tarefa escolhida .

*D. Metodologia de avaliação*

A avaliação do trabalho foi realizada a partir da aferição e comparação do tempo de duração de dez execuções para as modalidades propostas, quais sejam: execução serial (SER), paralela com *threads* (PTH) e paralela com processos (PPR). Em um primeiro momento sugeriu-se a avaliação do desvio padrão dos resultados em cada abordagem para avaliação de grandes perturbações do ambiente operacional no qual se processaram os testes. O *footprint* do sistema durante a execução também foi apresentado com base em gráficos que evidenciam a carga do processador durante a execução do pré-processamento de código fonte em cada abordagem proposta neste trabalho.

## Resultados e discussão

Esta seção se dedica à exploração dos resultados obtidos a partir da execução do processo de pré-processamento de código fonte.

*A. Análise de resultados com base na observação de séries*

Neste trabalho optou-se por fazer uma comparação das séries de aferições entre abordagens. Cada série teve seu desvio padrão calculado e os resultados foram 24.33, 5.91 e 10.50 para as abordagens ESE, PTH e PPR, respectivamente. Todos representam um baixo desvio padrão em que o coeficiente de variação, isto é a razão entre o desvio padrão e a média, é menor que um. Este fato garante a qualidade da nossa amostra e atesta que não há grande variabilidade dentro da amostra de cada abordagem.

*B. Análise gráfica*

A Fig. 1C ilustra a carga do processador durante a realização dos testes. Assim, é possível observar que a carga do processador é maior na abordagem PPR onde todos os processadores trabalham em processos diferentes que dividem os quatro núcleos físicos do processador durante a tarefa de pré-processamento. Com base na Fig. 1A e 1B, podemos verificar que existe uma subutilização do processador, nas abordagens SER e PTH, limitando sua carga de processamento a aproximadamente 1/4 de sua capacidade total.

O Gráfico 1 traz a plotagem de todos os resultados obtidos para todas as combinações (tentativa versus tempo de execução). É notória a superioridade da abordagem PPR em que todos os tempos de execução (assinalados em vermelho) são inferiores a quaisquer das tentativas para as outras abordagens SER e PTH.

## *Considerações finais*

Neste trabalho buscou-se evidenciar a diferença em termos de tempo de execução de uma mesma tarefa utilizando-se abordagens diferentes na gestão de processos e *threads* na linguagem de programação Python. Foi demonstrado que a abordagem PPR, baseada em execução simultânea de processos, é superior às demais abordagens que não conseguem fazer uso efetivo de todos os núcleos de um sistema *multicore*. Para trabalhos futuros propõem-se a utilização de outras linguagens de programação como Java ou C++, bem como a exploração de novas arquiteturas de hardware como sistemas com múltiplos processadores.

## Referências bibliográficas

ISSN: 1806-549X

Hennessy, J.L. et al. **Arquitetura de Computadores :** UMA ABORDAGEM QUANTITATIVA. Tradução da 5ª edição. Elsevier Editora Ltda. 2014.

Kamal, R. **Embedded Systems:** Architecture, Programming & Design. 3rd. McGraw Hill Education (India) Private Limited. 2015.

Kuhn, A. et al. **Semantic clustering :** Identifying topics in source code. Information & Software Technology. v. 49 , n. 3, pp. 230-243, 2007.

Paech, B. e Hubner, P. **Using Interaction Data for Continuous Creation of Trace Links Between Source Code and Requirements in Issue Tracking Systems.** Requirements Engineering: Foundation for Software Quality. Springer International Publishing. pp 291-307, 2017.

Phillips, D. et al. **Python :** Real-World Data Science. Editora Packt Publishing Ltd. ISBN 1786468417, 9781786468413, 2016, 1255p.

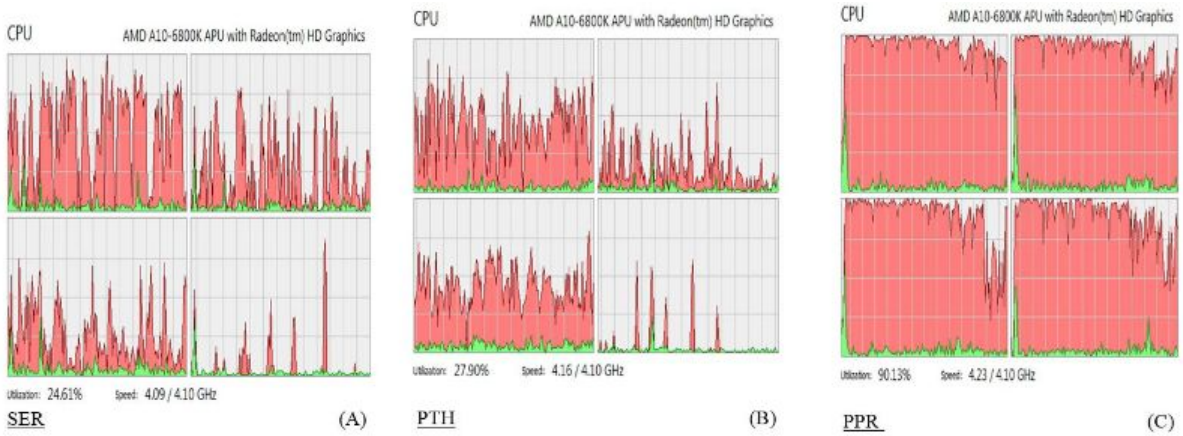


**Figura 1.** *Footprint* durante execução de cada abordagem

ISSN: 1806-549X

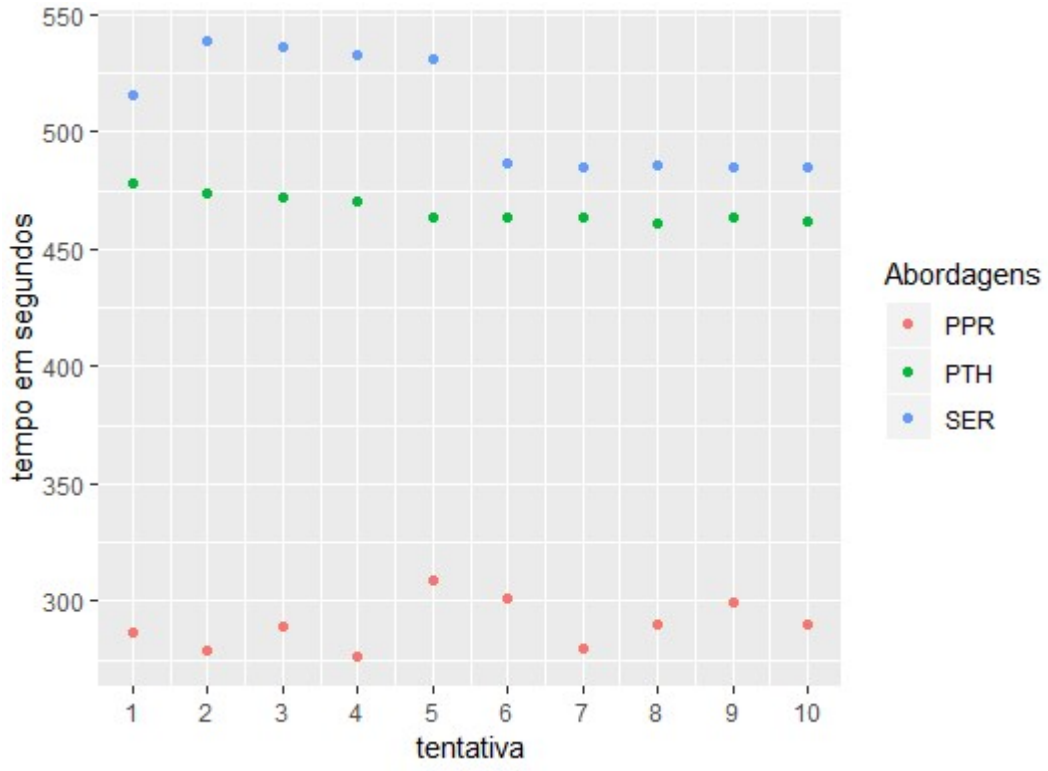


**Gráfico 1.** Tempo de execução do pré-processamento em cada abordagem