## Pequeno Escalonador por prioridades e créditos

Sistemas Operacionais (ACH2044)

Norton Trevisan Roman

Sistemas de Informação

Universidade de São Paulo

Lucas de Lyra Monteiro (15471435)

## 1. Pequena introdução

Para o desenvolvimento desta entrega foi utilizada a somente linguagem de programação python, cujo código e logs pode ser lido em sua íntegra, inclusive com seus versionamentos <u>neste repositório</u> (<a href="https://github.com/LucasdeLyra/scheduler">https://github.com/LucasdeLyra/scheduler</a>).

Devido a simplicidade do projeto este foi dividido em somente dois arquivos, auxiliar.py e escalonador.py, que contém códigos auxiliares de leitura de arquivos e ordenamentos diversos e o escalonar de fato pedido, respectivamente. O arquivo principal (escalonador.py) dispõe de somente duas classes: BCP, representando o Bloco de Controle de Processos e Escalonador, representando o escalonador que de fato interage com os BCP e, consequentemente, com os processos. A tabela de processos e as listas de processos prontos e processos bloqueados tornaram-se atributos no Escalonador. Provavelmente, caso a complexidade do trabalho fosse maior, seria ideal que elas tivessem suas próprias classes; como este não é o caso, são simplesmente objetos "list" do python.

## 2. Avaliação do sistema

Optei por testar o programa com 21 tamanhos de quantum diferentes, de 1 a 21 comandos (n\_com), visto que o sistema suporta no máximo 21 instruções por programa. Abaixo, pode-se observar a tabela com as métricas "Média de trocas", indicando a quantidade média de trocas de contexto e "Médias de instruções", indicando a quantidade média de instruções por quantum para os valores citados.

	Valor do n_com									
	1	2	3	4	5	6	7	8	9	10
Média de trocas	13,0	7,4	5,6	5	4,2	4	3,7	3,7	3,6	3,6
Média de instruções	1	1,75	2,35	2,67	3,15	3,39	3,65	3,65	3,75	3,75

	Valor do n_com										
	11	12	13	14	15	16	17	18	19	20	21
Média de trocas	3,5	3,5	3,5	3,3	3,3	3,3	3,3	3,3	3,3	3,3	3,2
Média de instruções	3,97	3,97	3,97	4,08	4,08	4,08	4,08	4,08	4,08	4,08	4,21

Analisando os dados acima é possível perceber que quanto maior o valor de n\_com, menor a média de trocas de contexto e maior a média de instruções por quantum. Contudo, somente com estes fatos não se pode concluir que quanto maior o n\_com melhor o nosso sistema é.

Ao averiguar mais profundamente percebe-se que temos faixas de valores em que as nossas métricas não se alteram, por exemplo, de 14 a 20 instruções por quantum manteve-se 3,3 trocas de contexto e 4,08 instruções por quantum. Este fato já nos dá indícios que aumentar o número de comandos de 14 para 20, pelo menos no caso deste sistema, não cause benefícios.

Ademais, é interessante avaliar a média de instruções comparado quanto ao número total de instruções disponíveis num quantum. Por exemplo, um valor de n\_com=21 nos dá o maior valor de instrução por quantum (4,21), entretanto, este valor de 4,21 equivale a somente 20% do máximo de instruções que poderíamos fazer por quantum, enquanto isso com n\_com=3 temos uma utilização de aproximadamente 78% do número máximo de instruções.

Outras estatísticas interessantes são o número de instruções de entrada e saída (23) e o número de operações COM (71). Infelizmente não consegui achar algum dado de entrada e saída para determinar se o sistema é CPU-Bound ou I/O-bound, mas é importante avaliar isto.

Além disso, não foi especificado o objetivo do sistema descrito, desta forma, fica-se prejudicada uma análise mais assertiva do quantum mais adequado. um sistema de computador pessoal (PC) precisa de um quantum pequeno para agradar o usuário médio, já um servidor ou uma máquina que processe arquivos em batch pode alocar um quantum maior para seus processos.

Assumindo que a operação COM seja consideravelmente gastosa\* (i.e, gaste muitos ciclos de clock) e também avaliando o disposto acima podemos chegar em algumas conclusões:

- Caso o sistema seja pensado para um computador pessoal, como Windows, Ubuntu ou MacOS, é interessante que o mesmo tenha um quantum baixo, de 3 a 7, garantindo que o usuário tenha uma sensação de que os processos estão todos sendo executados de forma síncrona, mesmo que tenhamos troca de contextos mais custosas.
- Caso o nosso sistema seja um servidor robusto, o melhor seria adotar valores altos, como 21 comandos por quantum, minimizando assim o overhead das trocas de contexto e maximizando as instruções por quantum.

\*Um computador atual consegue performar milhões de comandos em curtíssimos espaços de tempo, tendo um time-slice (quantum) entre 10 e 100 milissegundos, normalmente. Então estou supondo que COM é um comando que gasta *muito* tempo (ou que temos um computador muito ruim). Caso contrário, a análise acima faria pouco sentido visto a diferença de ordem de grandeza de duas dezenas de comandos para milhões de comandos.