Simulação de Escalonamento de Processos em Round Robin

Alexandre Araujo[1], **Gabriel Kurtz**[2]

[1]Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brazil

alexandre.henrique@acad.pucrs.br, gabriel.kurtz@acad.pucrs.br

Abstract. *This article proposes a software as a solution to simulate a process scheduler interacting with both operation system and CPU. The Round Robin algorithm is used to define the execution plan.*

Resumo. *Este artigo propõe uma solução em software para simular um escalonador de processos interagindo com ambos sistema operacional e processador. É utilizado o algorítimo Round Robin para definição do plano de execução dos processos.*

1. Introdução

O escalonamento de processos tem um papel fundamental na organização e distribuição de tarefas a serem executadas pelo processador, mantendo uma interação fluída entre o sistema operacional e seus periféricos. O objetivo deste estudo é desenvolver uma aplicação da qual seja capaz de simular o funcionamento parcialmente completo de um escalonador e destacar seu desempenho e eficiência perante a instruções de casos de uso. As etapas do método de organização e direcionamento dos processos serão explicadas detalhadamente e abrangem desde a entrada de novos processos, até a troca de contexto, saída para processamento e IO, assim como o reordenamento dos processos enfileirados de acordo com sua prioridade.

Para criação do software, foi utilizada a linguagem Python, que, por sua fácil utilização, baixa fidelidade a tipos, bibliotecas nativas e recursos poderosos, contribuiu para um desenvolvimento rápido e de fácil leitura.

2. Comportamento do Escalonador

2.1. Leitura de cenários

Cenários são instruções pré-estabelecidas que formam o comportamento simulado do sistema operacional interagindo com escalonador de processos. Nele são definidos a quantidade de processos a serem inseridos, a fatia de tempo em que cada processo permanece executando no mesmo processador, o tempo de chegada e tempo de saídas de cada processo, assim como sua prioridade também. Por definição, o simulador entende a seguinte estrutura de dados de entrada, onde os valores são separados apenas por espaços e quebras de linha:

```
5
7
1  10  1  5  9
1  12  1
35  15  2  4  8  12
35  15  1
35  8  1
```

Onde:

- Primeira linha: número de processos a serem lidos
- Segunda linha: tamanho em unidade de tempo T, da fatia de tempo.
- Terceira linha em diante: processos separados por:
  - Tempo de chegada
  - Tempo de execução
  - Prioridade
  - Tempo de acesso a operações de entrada e saída

2.2. Novos processos

O primeiro passo na rotina de execução do escalonador é identificar a entrada de novas tarefas. Essa verificação é feita através de uma comparação entre o contador de tempo atual e o tempo de cada processo que pertence a fila pré-estabelecida de entrada. Uma vez detectados, os processos são distribuídos entre as filas de execução de acordo com as suas prioridades.

2.3. Troca de contexto

Optamos por realizar uma troca de contexto sempre que um processo está sendo executado e ocorre a troca por outro processo, representando o tempo que o processador leva para gravar as informações relevantes do estado do processo atual e também carregar o outro processo.

Isto pode ocorrer em diferentes situações:

- Quando chega um processo com prioridade mais alta que o atual (o processo com maior prioridade passa a ser executado, reparando que a maior prioridade é representada por um menor número)
- Quando um processo atinge sua fatia de tempo. Neste caso o processo volta para a fila, caso haja, de onde um novo processo é chamado (caso não haja outros processos na mesma prioridade, ocorre a troca de contexto e volta o mesmo processo)
- Quando um processo passa para a fila de Entrada/Saída

Seguindo esta lógica, optamos que, nos casos em que um processo é chamado por um processador vazio, não ocorre troca de contexto, de modo a ficar semelhante com o exemplo contido no enunciado. Para os casos em que ocorre Entrada/Saída, isto não interfere no tempo que leva para o processo voltar (ainda consideramos que um processo que inicia E/S no tempo x retorna ao processador pelo menos em x+5 embora a segunda troca de contexto nao seja impressa no gráfico).

2.4. Execução

A CPU (central processing unit) é a cerne da escalonador. É nela onde ocorre a execução das tarefas oriundas da fila de prioridade. Este processamento unitário, entretanto, é limitado por T unidades de tempo, para que, assim, haja um plano de execução melhor distribuído entre os processos de mesma prioridade. Além do limite de tempo, o processo pode ter que parar sua execução por ter sido finalizado ou também por ter que sair para resposta de E/S.

2.5. Respostas de Entrada e Saída (E/S)

Alguns processos podem possuir, além do Tempo de Execução, requerimentos de Entrada e Saída. Ficou definido, para fins deste exercício, que cada resposta de E/S leva quatro unidades de tempo. Cada processo pode ter ou não uma lista de E/S, definida no arquivo de input de dados, indicando os momentos em que devem entrar na fila de E/S. Ou seja, se um processo tem uma lista [5, 10], ele deve parar de processar e passar para a fila de E/S ao concluir o quinto e o décimo períodos de processamento de CPU.

Definimos, de forma semelhante aos exemplos, que o primeiro processo de E/S ocorre no mesmo momento que o último de CPU, ficando o processo fora da CPU por três unidades posteriores. Ao retornar, caso o processador esteja vazio, optamos por levar mais uma unidade de tempo, de modo que um único processo ficaria ocioso por 4 unidades, novamente como nos exemplos fornecidos. No entanto, caso o processador esteja vazio, optamos por não marcar Troca de Contexto. Caso haja algum processo rodando, o processo que voltou da E/S passa novamente por um ordenamento. Consideramos tambem que a chamada de E/S não zera o contador de troca de contexto daquele processo. Todas estas decisões foram tomadas porque faziam mais sentido no momento, e podem ser revertidas com pequenas mudanças de código.

2.6. Reordenamento de filas

O escalonador trabalha com duas filas distintas: READY e PRIORITY. A primeira comporta o conjunto de todas as tarefas que estão prontas para executar, porém com prioridade baixa. Enquanto que a segunda aloca os processos que possuem a prioridade mais alta entre os processos prontos.

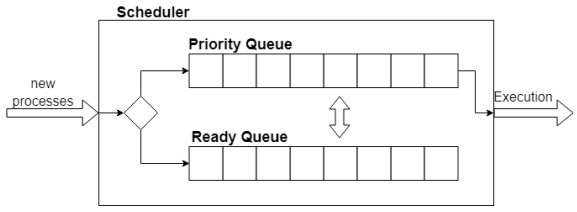

Figura 1. Diagrama geral do escalonador

Conforme novos processos chegam, passam por uma fila de ARRIVALS, indo então para a fila READY. A cada ciclo (unidade de tempo), o algoritmo checa se algum processo que chegou tem a prioridade maior ou igual à mais alta e toma o ordenamento correto: no caso de chegar uma prioridade mais alta, retorna o processo em execução e toda a fila PRIORITY para ready, e então busca em READY todos os processos com a nova maior prioridade, colocando o primeiro em execução e os demais na fila PRIORITY.

Caso os novos arrivals tenham, no máximo, a mesma prioridade que a mais alta, basta adicioná-los à fila PRIORITY e manter os demais na fila READY.

3. Medidas Relevantes de Processos

3.1. Turn Around Time (TAT)

O Turn Around Time representa o tempo que o processo levou para ser concluído depois de ter ficado pronto. Ou seja, após o Arrival Time, quanto tempo decorreu até o seu Conclusion Time. Para caculá-lo, portanto, a formula é: TAT = CT - AT. Como no nosso exemplo ficou definido no enunciado que o processo chega em x e inicia em x+2, o tempo de conclusão considera este fato.

3.2. Waiting Time (WT)

Representa quanto tempo o processo ficou esperando. Ou seja, é a soma de todos os ciclos que o processo, após estar pronto (ter passado seu Arrival Time), não foi processado pela CPU por qualquer razão (não ser o primeiro da fila ou estar em E/S).

Tendo calculado o Turn Around Time, torna-se fácil calcular o Waiting Time: WT = TAT - BT. Ou seja, basta subtrair do Turn Around Time o Burst Time pois, ao final do processo, seu Burst Time representa o tempo que ele ficou na CPU.

3.3. Response Time (RT)

O tempo de resposta representa quanto tempo o processo levou (após seu AT) para dar uma primeira resposta. Para este exercício, consideramos que a primeira resposta ocorre na primeira unidade de tempo em que o processo é tratado pelo processador.

Para calculá-la, iniciamos com uma variável negativa (-1). Sempre que o processo executa, verifica se a variável do RT está negativa. Caso esteja, altera para o valor de espera atual (Tempo - Arrival Time). Como este valor será obrigatoriamente positivo (ou igual a zero), não será mais alterado, mantendo o valor de espera no momento que foi gerada a primeira resposta.

4. Conclusão

Acreditamos ter conseguido produzir um algoritmo bastante razoável e, dentro dos parâmetros estabelecidos, correto. Durante uma fase bastante avançada da produção, percebemos que teria sido melhor utilizar uma fila para cada prioridade, pois assim conseguiríamos simplificar bastante a administração dos processos.

No entanto, naquela altura, era muito mais viável implementar o que faltava naquele modelo apesar de mais trabalhoso.

Apesar disso, temos confiança que o nosso script é capaz de resolver o problema com uma boa precisão e que foi um aprendizado bastante importante.

Nos exemplos em que havia dados disponíveis para comparar, nosso algoritmo produziu cálculos das métricas de processos iguais aos exemplos ou, nos casos em que houve pequenas diferenças (relacionadas especialmente à fila de E/S ou então ao tempo extra por iniciar em t=x+2 processos que chegam em t=x), é possível identificar a razão comparando a linha do tempo de ambos os processadores e compreendendo as diferenças de interpretação de projeto contidas neste artigo.