Relatório - Escalonamento de Processos

Allan G. A. Oliveira[1], **Joan A. Medeiros**[2]

[1]Centro de Ensino Superior do Seridó (CERES) – Caicó – RN – Brazil

[2]Departamento de Computação e Tecnologia
Universidade Federal do Rio Grande do Norte (UFRN) – Caicó – RN – Brazil

allan.oliveira.110@ufrn.edu.br, joan.medeiros.130@ufrn.edu.br

Abstract. *This work is a report about the results of 250 simulations involving five scheduling algorithms. Among these algorithms, four of them were produced: the FCFS (First-Come First-Served), the SJF (Shortest Job First), the RR (Round Robin) and the priority one (PRIO). In order to prepare this report, in addition to the implementations of the algorithms, a battery of tests was carried out, involving 50 rounds for each of the algorithms, varying only the input parameters, thus totaling 250 simulations. In the report, three graphs were built, one for Average Waiting Time (total), one for Average Waiting Time (CPU Bound) and one for Average Waiting Time (I/O bound).*

Resumo. *Este trabalho se trata de um relatório a respeito dos resultados de 250 simulações envolvendo cinco algoritmos de escalonamento. Entre estes algoritmos, quatro deles forão produzidos: o FCFS (First-Come First-Served), o SJF (Shortest Job First), o RR (Round Robin) e o de prioridades (PRIO). Para elaborar este relatório foi realizado além das implementações dos algoritmos, uma bateria de testes, envolvendo 50 rodadas para cada um dos algoritmos, variando-se apenas os parâmetros de entrada, totalizando assim 250 simulações. No relatório, foram construídos três gráficos, um para o Tempo Médio de Espera (total), um para o Tempo médio de espera (CPU Bound) e um para o Tempo médio de espera (I/O bound).*

1. Discussão dos algoritmos

Os algoritmos foram implementados tendo como base o Simulador de Escalonador de curto prazo do Renato Cunha, o qual foi construído utilizando a linguagem de programação C. O programa contém diversas funções que permitem a simulação do funcionamento de um escalonador em um Sistema Operacional (SO), simulando interrupções e realizando a realocação de processos entre as filas utilizadas pelo SO para organização dos processos. Para isso, o algoritmo conta com diversas funções, mas a principal é a função que realiza o escalonamento, pois é ela que gerencia os processos que estão em execução, fazendo a manipulação das filas de processos aptos e bloqueados.

Como dito na seção de resumo, os algoritmos implementados forão o FCFS (First-Come First-Served), o SJF (Shortest Job First), o RR (Round Robin) e o de prioridades (PRIO). O primeiro algoritmo a ser implementado foi o FCFS, o qual é bastante parecido com o algoritmo base, diferindo apenas na sua função de escalonamento, onde é utilizado apenas uma fila de processos aptos. Neste algoritmo, os processos são executados de forma sequencial obedecendo a ordem de criação de cada processo, e quando ocorre

uma interrupção, o processo é colocado na fila de bloqueados e quando se tornar apto novamente, o mesmo é realocado no final da fila de aptos.

O segundo algoritmo a ser implementando foi o Round Robin. Este algoritmo também foi relativamente fácil de se implementar, semelhante ao FCFS este também utiliza apenas uma fila de processo aptos, a diferença é que se trata de um algortimo preemptivo, sendo necessário realizar as trocas de contexto quando um determinado tempo de QUANTUM se esgota. O QUANTUM é o critério que o Round Robin utiliza para interromper a execução de um processo em benefício de outro, ou seja, o processo perde o processador e o próximo processo da fila ganha. Dessa forma, este algoritmo apresenta um comportamento bastante semelhante ao FCFS já que os processos seguem uma espécie de sequência, sendo alternados ao longo da execução por um valor de QUANTUM.

O terceiro algoritmo a ser implementado foi o de prioridades. Este algoritmo apresentou uma certa complexidade, devido utilizar duas filas de processos aptos. Na sua implementação é necessário que uma das filas exerça prioridade sobre a outra, assim, os processos que estão na fila com menor prioridade só podem executar quando todos os processos da fila de maior prioridade terminarem sua execução. Dessa forma, surgiu um questionamento a respeito dos processos bloqueados, onde se foi questionado para onde esses processos deveriam ser alocados quando se tornassem aptos novamente. Diante disso, foi julgado válido realizar uma alteração na estrutura de dados do algoritmo base, criando-se um campo para armazenar a prioridade do processo, permitindo identificá-lo na fila de bloqueados e realocá-lo em sua fila original de criação.

Finalizando as implementações, foi implementado o SJF. Este foi o algoritmo que apresentou maior complexidade na sua construção. Esta complexidade se deu devido o algoritmo implementar uma política, cuja ordem de execução dos processos ocorre com base no seu tempo de execução, impondo que os processos sejam executados em ordem crescente do tempo de execução. Dessa forma, após a criação de todos os processos, foi necessário implementar um procedimento que realizasse uma ordenação dos processos, de forma que o primeiro a executar fosse o processo de menor tempo. Para realizar este procedimento, foram utilizadas filas auxiliares, onde uma delas continha os processos na ordem em que foram criados e a outra continha os processos na ordem do tempo de execução de forma crescente.

2. Análise das simulações

As simulações foram executadas para os cinco algoritmos de escalonamento. Para cada algoritmo foi realizado 10 simulações para cada x quantidade de processos, sendo a quantidade de processos: 10, 20, 30, 40 e 50. Dessa forma, cada algoritimo foi executado 50 vezes, totalizando um montante de 250 simulações ao todo.

A cada simulação foi registrado o Tempo Médio de Espera (total), o Tempo médio de espera (CPU Bound) e o Tempo médio de espera (I/O bound). De posse desses valores, foi calculado para cada par “algoritmo x quantidade de processos” o valor médio das amostras do Tempo Médio de Espera (TME), assim como a variância entre as amostras do TME. Abaixo segue as imagens dos gráficos:

2.1. Grafico 1 - Tempo médio de espera (total)

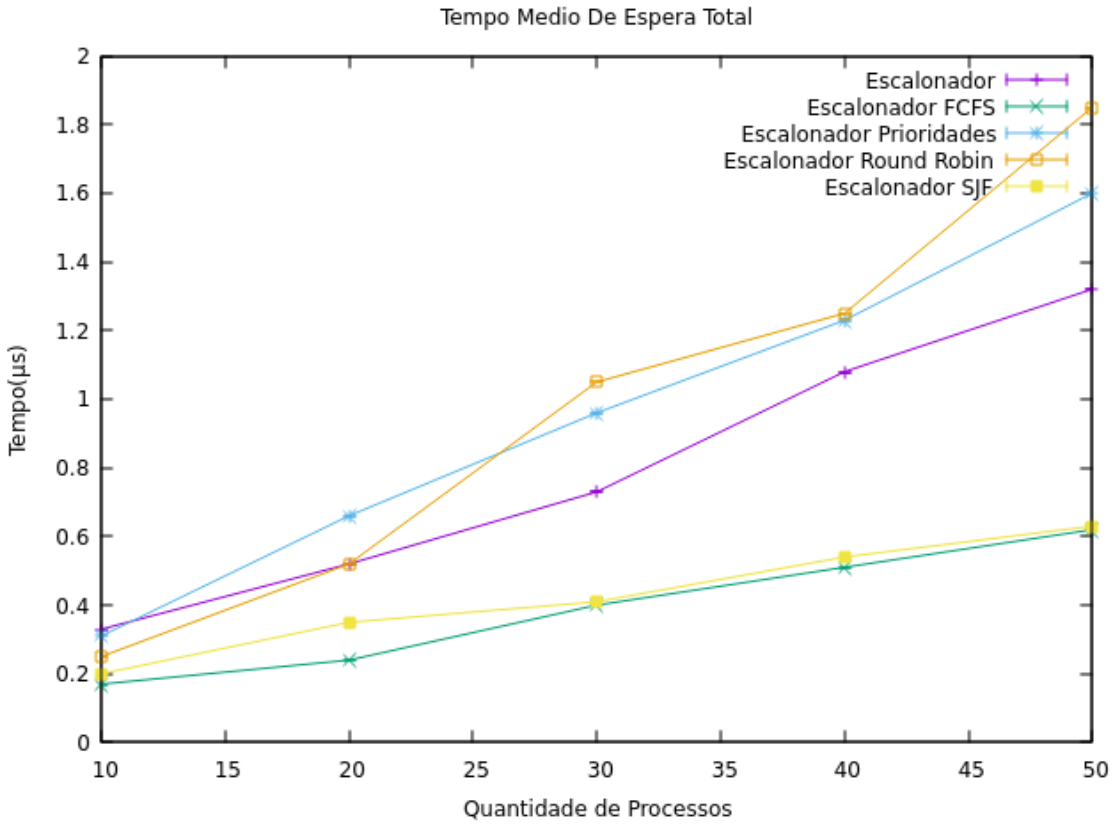

Figura 1. Tempo Médio de Espera Total

Como pode ser observado no gráfico acima, conforme o número de processos aumenta, o Tempo Médio de Espera (TME) também aumenta, e isso é válido para ambos os algoritmos. Também podemos observar que os algoritmos que possuem o menor (TME) são os algoritmos FCFS e SJF, possuindo uma diferença entre si relativamente pequena. Já os que apresentaram o maior (TME) foram o RR e o PRIO. Este comportamento pode ocorrer por diversos fatores, como por exemplo o tamanho do QUANTUM, os tempos de execução de cada processo, interrupções do sistema. Todos esses fatores influenciam no (TME), e além disso, devemos levar em consideração que todas as simulações foram feitas utilizando tempos de execução randômicos diferentes para cada um dos algoritmos.

2.2. Grafico 2 - Tempo médio de espera (CPU bound)

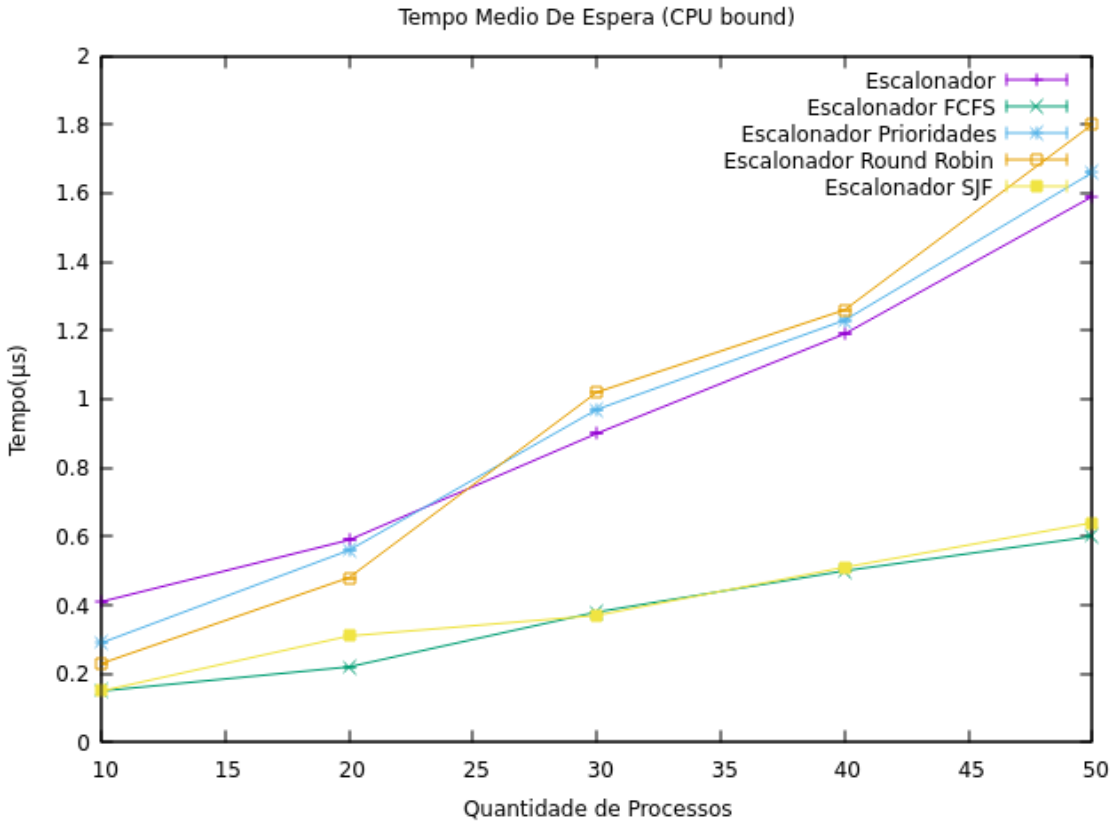

Figura 2. Tempo Médio de Espera (CPU bound)

No gráfico 2 percebemos que os algoritmos FCFS e SJF, assim como no gráfico 1, continuam possuindo os menores (TME), enquanto o RR e o PRIO continuam com os maiores (TME). Aqui voltamos a mesma discussão da análise no gráfico 1, as simulações foram feitas com valores randômicos, então pode ocorrer que durante os testes, o número de processos classificados como CPU bound sejam diferentes entre os algoritmos.

Um ponto em importante que podemos destacar é que os algoritmos não preemptivos (FCFS e SJF) apresentam (TMEs) bastante semelhantes, o que também ocorre para os algortimos preemptivos (RR e PRIO). Diante disso, podemos supor que o comportamento do gráfico está totalmente associado a existência ou não de preempção.

2.3. Grafico 3 - Tempo médio de espera (I/O bound)

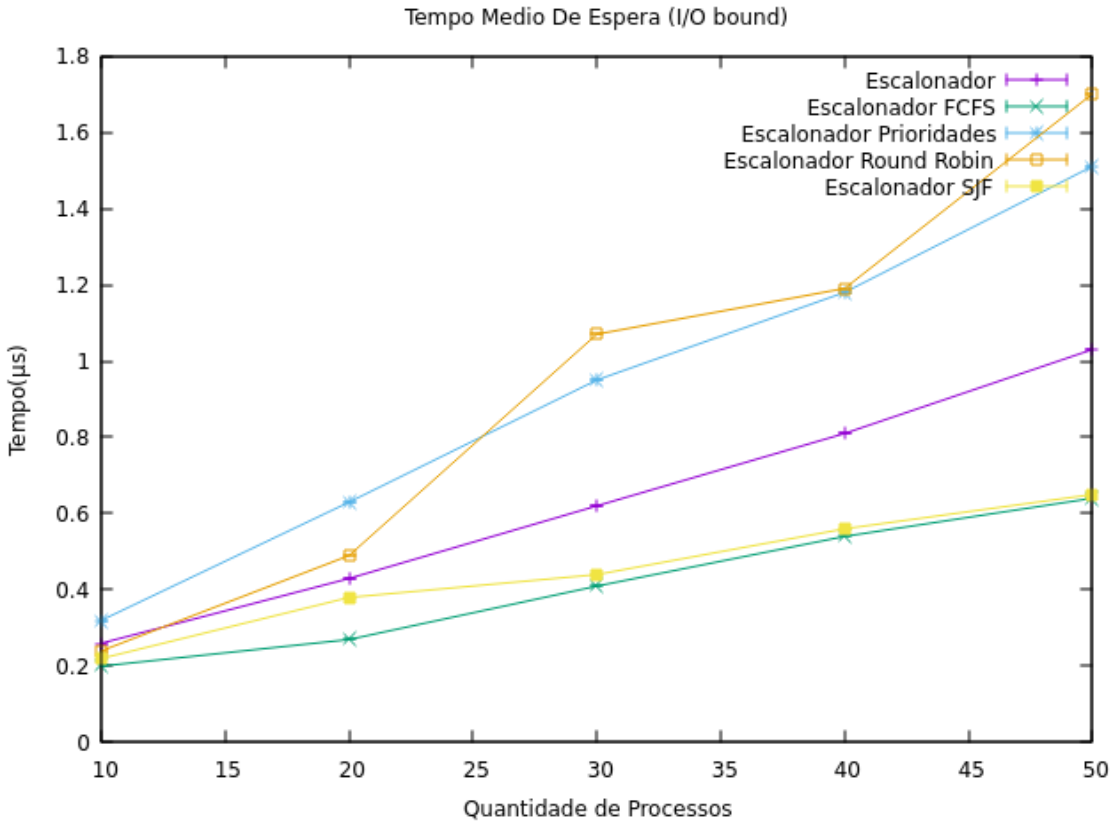

Figura 3. Tempo Médio de Espera (I/O bound)

Finalizando a análise, temos o gráfico 3 que apresenta os resultados a respeito do Tempo Médio de Espera para processos I/O bound, que ao ser comparado com os demais gráficos, o mesmo não apresenta grandes divergências. Um ponto em comum presente em ambos os gráficos, é que os algoritmos FCFS e SJF sempre apresentam os menores (TME), enquanto o RR e o PRIO continuam com os maiores (TME). Como já discutido anteriormente, este comportamento pode estar associado a diversos fatores que influenciam ao gráfico possuir esse comportamento, como por exemplo a existência de preempção, o número de trocas de contexto entre outros.