Avaliação de desempenho da biblioteca

Foi avaliado o tempo de execução do programa exemplo para diferentes números de processadores virtuais, e diferentes valores de entrada para o Fibonacci. Foi utilizado o algoritmo de escalonamento FCFS.

Abaixo está a tabela apresentando os resultados encontrados, onde a coluna representa a entrada usada no Fibonacci, e a linha representa o número de processadores utilizados.

	7	**14**	**21**	**28**	**35**
3	0:0.002	0:0.003	0:0.054	0:1.929	0:59.541
6	0:0.002	0:0.004	0:0.052	0:2.102	1:12.614
9	0:0.003	0:0.005	0:0.075	0:1.841	1:22.966
12	0:0.002	0:0.005	0:0.094	0:2.093	1:35.484
15	0:0.002	0:0.011	0:0.079	0:2.901	1:34.556

Foram escolhidos 5 valores para o número de processadores virtuais (de 3 à 15, utilizando múltiplos de 3), e 5 valores para entrada do Fibonacci (de 7 à 35, utilizando múltiplos de 7).

No geral, foi observado que números maiores de processadores apresentaram resultados inferiores a números menores. Isso pode ser explicado pelo fato de que quantidades maiores de processadores trazerem uma complexidade maior no seu gerenciamento, que pode superar a economia de tempo provocada pelo multiprocessamento. Os resultados podem ser influenciados por processos externos do computador.

Comparação entre algoritmos de escalonamento

Foram implementados 3 algoritmos de escalonamento na biblioteca: FCFS, que consiste em um fila simples; SJF, que escolhe sempre o processo de menor custo; e PRIOc, que determina o próximo processo de acordo com sua complexidade.

O programa exemplo do Fibonacci não faz distinção de prioridade entre os processos, em razão disso, foram comparados apenas os algoritmos FCFS e SJF. Os resultados dessa avaliação estão apresentados na tabela abaixo.

	7	**14**	**21**	**28**	**35**
FCFS	0:0.003	0:0.006	0:0.063	0:1.472	0:55.497
SJF	0:0.001	0:0.003	0:0.034	0:0.598	0:22.092

Foram escolhidos os mesmos 5 valores de entrada do Fibonacci da análise anterior para essa comparação. O número de processadores virtuais foi definido como 2 para todas as execuções.

Conforme a tabela, é possível observar que o algoritmo SJF apresentou um desempenho melhor que o FCFS em todos os casos analisados. O algoritmo SJF perde desempenho por percorrer a lista de processos sempre que um novo processo é buscado, enquanto o FCFS tem complexidade constante para essa tarefa. Ainda assim, esse aumento de complexidade não supera os ganhos que esse algoritmo traz à biblioteca. Os resultados podem ser influenciados por processos externos do computador.