Relatório Técnico do Trabalho I da Unidade II de S.O.

Autores: Jônatas Câmara dos Santos & Pedro Henrique Ribeiro Alves
Data: 09/08/2024

Resumo

Este artigo aborda a implementação dos algoritmos de escalonamento de processos em sistemas operacionais, com foco na análise de desempenho comparativa. O estudo compara os algoritmos de escalonamento First-In, First-Out (FIFO), Shortest Job First (SJF), Longest Job First (LJF), Prio_Static (prioridades estáticas) e Prio_Dynamics (prioridades dinâmicas) em termos de métricas de desempenho como Tempo Médio de Espera (TME), com valor médio e variâncias das amostras do TME. Os resultados foram obtidos através de testes, onde variáveis como número de execuções e variação nas prioridades foram manipuladas para avaliar a eficiência dos algoritmos.

Palavras-chave: Escalonamento de Processos, Análise de performance comparativa entre os algoritmos.

Introdução

O escalonamento de processos é essencial para o desempenho dos sistemas operacionais, pois define como a CPU distribui seu tempo entre os processos. O objetivo é otimizar a eficiência e reduzir o tempo de espera.

Existem diferentes algoritmos para escalonamento, cada um com suas vantagens e desvantagens. Alguns algoritmos atendem processos por ordem de chegada, enquanto outros priorizam processos com base em seu tempo de execução ou prioridade. Este estudo compara esses algoritmos usando simulações para medir o tempo de espera, o tempo de retorno e a utilização da CPU.

Fundamentação Teórica:

FIFO (First-In, First-Out)

Conceito Básico:
O algoritmo FIFO é um dos métodos mais simples de escalonamento de processos. Ele segue o princípio de que o processo que chega primeiro é o primeiro a ser atendido. É semelhante a uma fila de atendimento, onde o primeiro a entrar é o primeiro a sair.

Funcionamento:

- **Entrada de Processos:** Os processos são adicionados à fila na ordem em que chegam.

- **Execução:** A CPU executa os processos na ordem em que estão na fila. O processo que está na frente da fila é escolhido para execução.
- **Objetivo:** Garantir que o processo mais antigo na fila seja executado primeiro, sem considerar o tempo restante ou prioridade.

SJF (Shortest Job First)

Conceito Básico:

O SJF prioriza a execução dos processos com o menor tempo de execução estimado. O objetivo é reduzir o tempo médio de espera dos processos na fila.

Funcionamento:

- **Entrada de Processos:** Os processos são ordenados com base em seus tempos de execução estimados.
- **Execução:** A CPU sempre escolhe o processo com o menor tempo de execução restante.
- **Objetivo:** Minimizar o tempo médio de espera e o tempo médio de resposta ao executar primeiro o processo que levará menos tempo para terminar.

LJF (Longest Job First)

Conceito Básico:

O LJF é o oposto do SJF, priorizando processos com o maior tempo de execução estimado. É menos comum, pois pode aumentar o tempo de espera para processos menores.

Funcionamento:

- **Entrada de Processos:** Os processos são ordenados com base em seus tempos de execução estimados, mas o processo mais longo é escolhido primeiro.
- **Execução:** A CPU executa o processo com o maior tempo de execução restante.
- **Objetivo:** Priorizar a execução de processos que ainda têm muito tempo para concluir, o que pode aumentar o tempo médio de espera, mas pode ser útil em algumas situações específicas.

PRIO Static (Priority Scheduling Estático)

Conceito Básico:

No escalonamento de prioridade estática, cada processo é atribuído uma prioridade fixa. A CPU é alocada ao processo com a maior prioridade.

Funcionamento:

- **Entrada de Processos:** Cada processo recebe uma prioridade que não muda durante sua execução.
- **Execução:** O processo com a maior prioridade é escolhido para execução. Se vários processos têm a mesma prioridade, eles são escalonados por FIFO.

- **Objetivo:** Oferece uma seleção de processo baseada em uma probabilidade ajustada dinamicamente, com preferência por filas com maior prioridade.

PRIO Dynamic (Priority Scheduling Dinâmico)

Conceito Básico:
No escalonamento de prioridade dinâmica, as prioridades dos processos podem mudar ao longo do tempo. As prioridades podem ser ajustadas com base em critérios como tempo de espera ou necessidade de recursos.

Funcionamento:

- **Entrada de Processos:** Cada processo começa com uma prioridade inicial, que pode ser ajustada durante a execução.
- **Execução:** A CPU é alocada ao processo com a maior prioridade dinâmica. As prioridades podem aumentar ou diminuir com o tempo, dependendo das políticas de ajuste.
- **Objetivo:** Oferece uma seleção de processo baseada em uma probabilidade ajustada dinamicamente, com preferência por filas com maior prioridade.

Metodologia

Para a construção e análise comparativa dos algoritmos de escalonamento de processos, seguimos os seguintes passos metodológicas:

1. Configuração dos Ambiente de Teste

- **Ferramentas e Tecnologias**: Utilizamos uma linguagem de programação de alto nível para implementar o simulador de escalonamento de processos. O ambiente de teste foi configurado para variar o número de processos, tempos de execução e prioridades, permitindo uma análise abrangente dos algoritmos.
- **Cenários de Carga de Trabalho**: Definimos diversos cenários de carga de trabalho, ajustando parâmetros como tempo de chegada dos processos, duração das execuções e prioridades atribuídas, para testar a eficácia de cada algoritmo em diferentes condições.

2. Métricas de Desempenho

- **Algoritmos Selecionados**: Implementamos os algoritmos FIFO, SJF, LJF, Prio_Static e Prio_Dynamic no simulador, assegurando que cada um fosse fiel às suas especificações teóricas.
- **Testes Preliminares**: Realizamos testes preliminares em cada algoritmo para verificar sua correta implementação e ajustar quaisquer discrepâncias iniciais.

3. Execução dos Testes

- **Execução dos Cenários**: Executamos cada algoritmo nos cenários de teste previamente definidos, garantindo múltiplas execuções para assegurar a consistência dos resultados.
- **Monitoramento**: Durante a execução, monitoramos o comportamento dos processos e a utilização da CPU para verificar a operação correta dos algoritmos e identificar possíveis anomalias.

4. Análise dos Resultados

- **Métricas de Desempenho**: Coletamos dados das principais métricas de desempenho, incluindo Tempo Médio de Espera (TME), Tempo de Retorno e Utilização da CPU.
- **Análise Estatística**: Calculamos os valores médios e variâncias das amostras do TME para cada algoritmo em diferentes cenários, comparando os resultados para identificar diferenças significativas e padrões de desempenho.

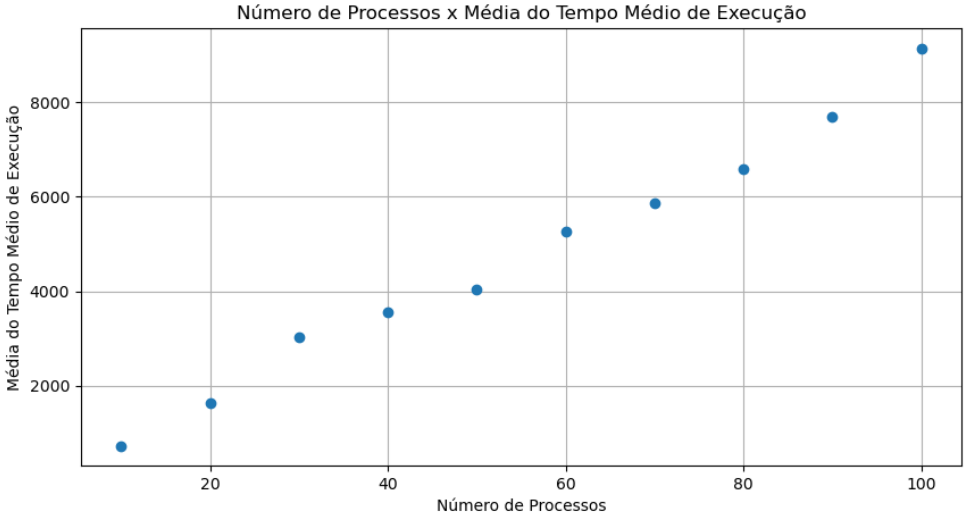
5. Validação dos Resultados

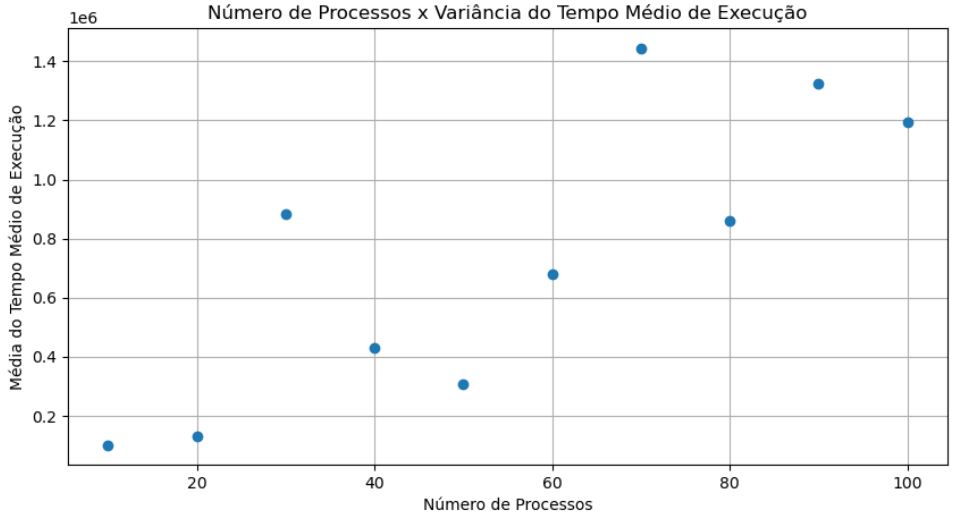
- **Precisão e Confiabilidade**: Validamos a precisão e confiabilidade dos resultados por meio de testes adicionais e validação cruzada com outros estudos.
- **Revisão e Correção**: Revisamos os dados coletados para identificar e corrigir inconsistências, garantindo a integridade dos resultados.
- **Documentação**: Compilamos os resultados e análises em um relatório técnico, apresentando os dados de forma clara e visual para facilitar a compreensão e interpretação dos resultados.

Resultados

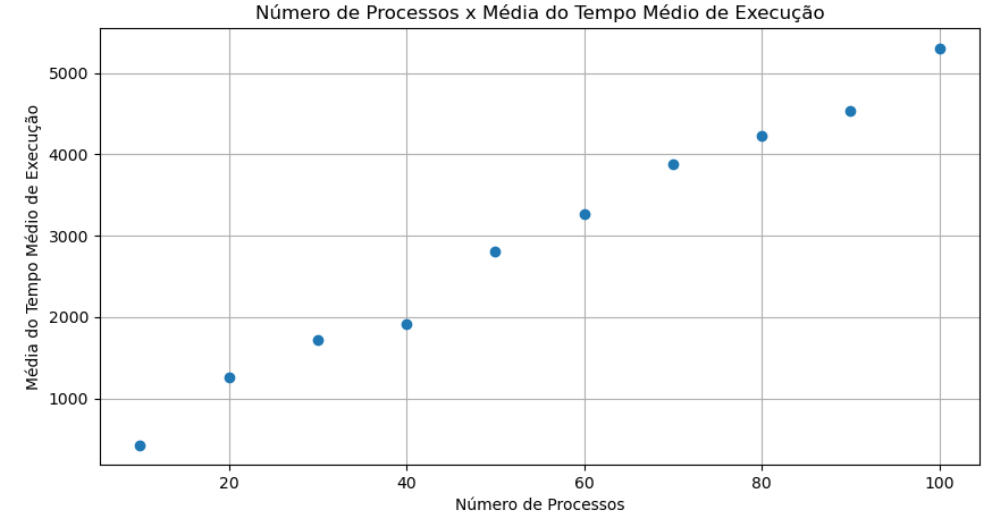
Com base na metodologia descrita, foram obtidos resultados variados para cada algoritmo analisado.

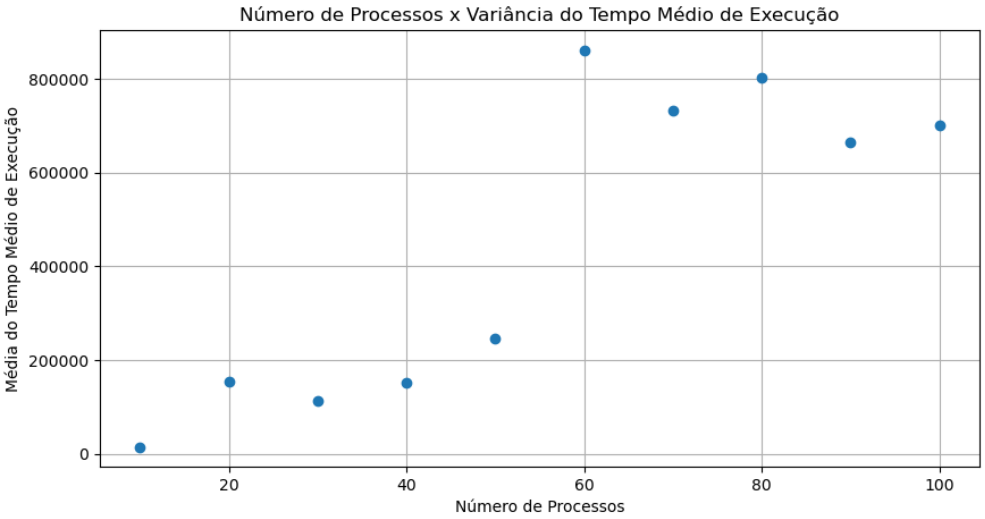
O algoritmo FIFO apresentou um tempo médio de espera (TME) elevado em cenários com alta variação no tempo de chegada dos processos, devido à sua falta de priorização, resultando em longos tempos de espera para processos que chegam posteriormente. Além disso, a variância do TME nas amostras do FIFO foi significativa, refletindo a inconsistência do desempenho em cenários com diferentes cargas de trabalho. Embora sua simplicidade de implementação seja uma vantagem, a eficiência é comprometida em situações de alta concorrência, com grandes flutuações no TME.



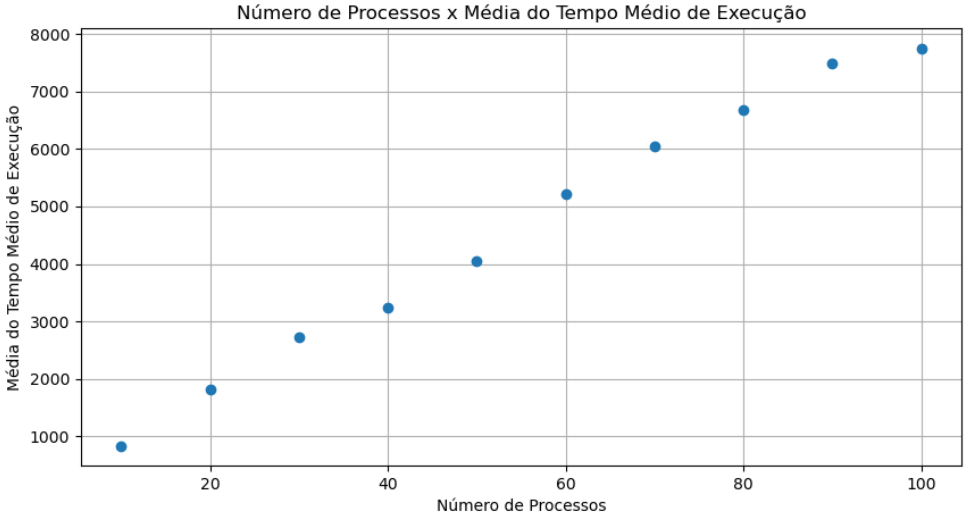


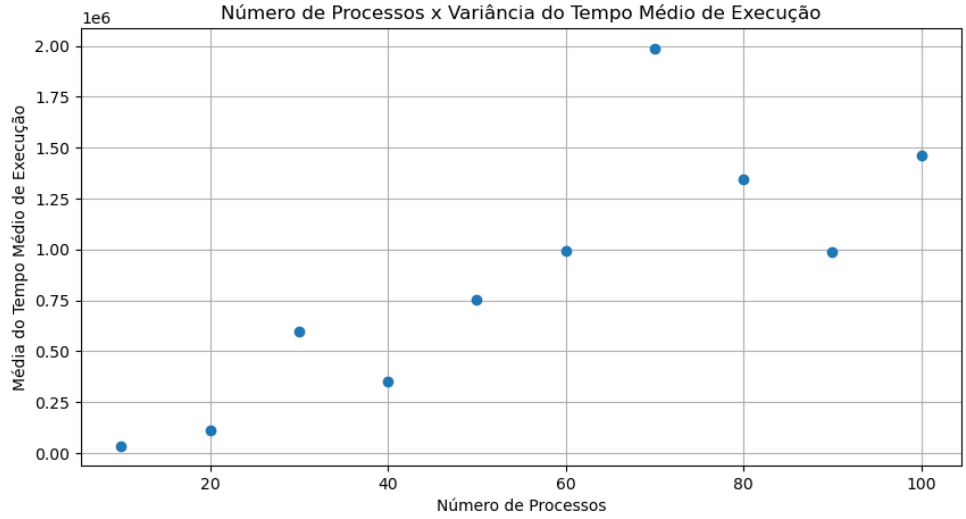
O algoritmo SJF se destacou por obter o menor TME, especialmente em cenários com significativa variação no tempo de execução dos processos, devido à sua priorização de processos curtos. A variância do TME entre as amostras do SJF foi relativamente baixa, indicando um comportamento mais previsível e consistente em diversas condições de carga. Isso torna o SJF uma escolha eficaz para reduzir o TME e melhorar a responsividade do sistema, embora seja necessário prever ou estimar o tempo de execução dos processos, o que pode ser um desafio em alguns casos.



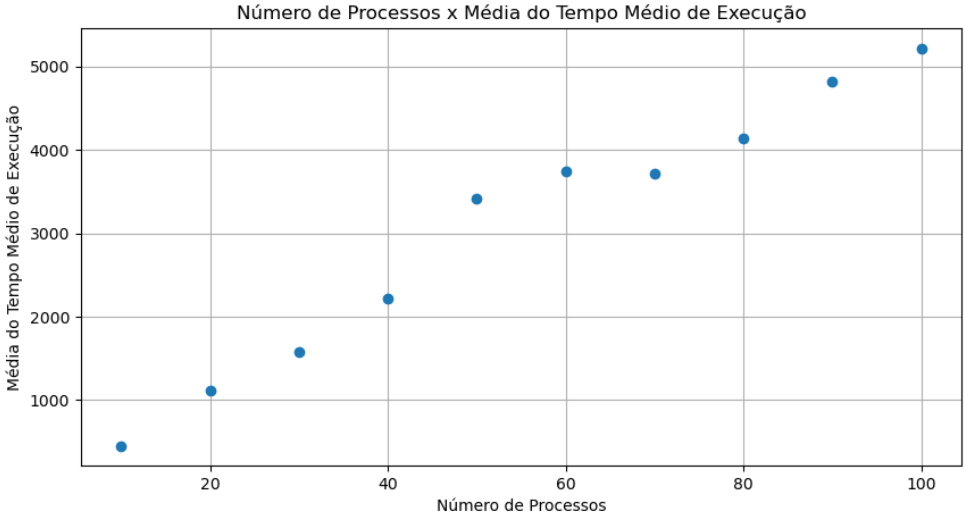


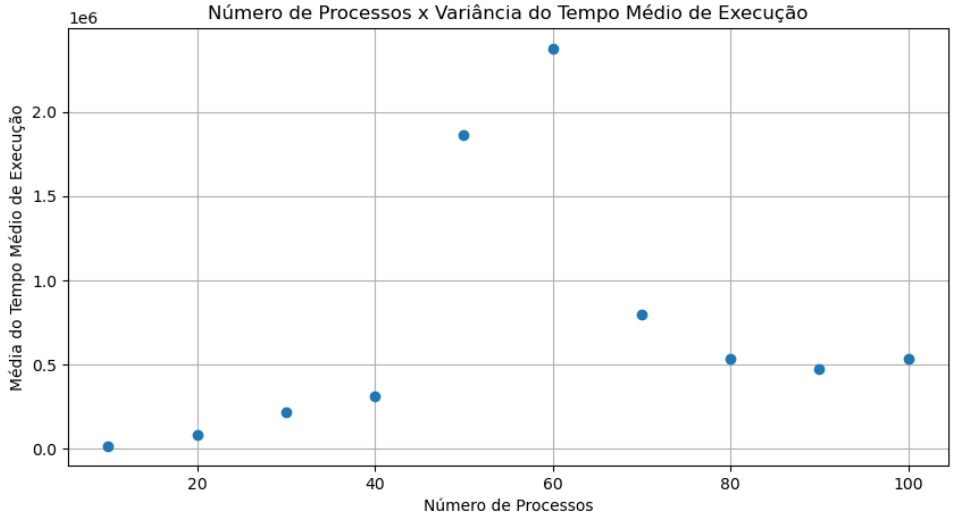
Por outro lado, o algoritmo LJF, que prioriza processos mais longos, resultou em um TME mais alto, aumentando o tempo de espera para processos mais curtos. A variância do TME no LJF foi a maior entre os algoritmos analisados, o que evidencia um desempenho mais errático e dependente das condições específicas da carga de trabalho. Apesar de poder evitar o *starvation* de processos longos em situações específicas, o LJF geralmente resulta em desempenho inferior e inconsistente, com grandes variações no TME.



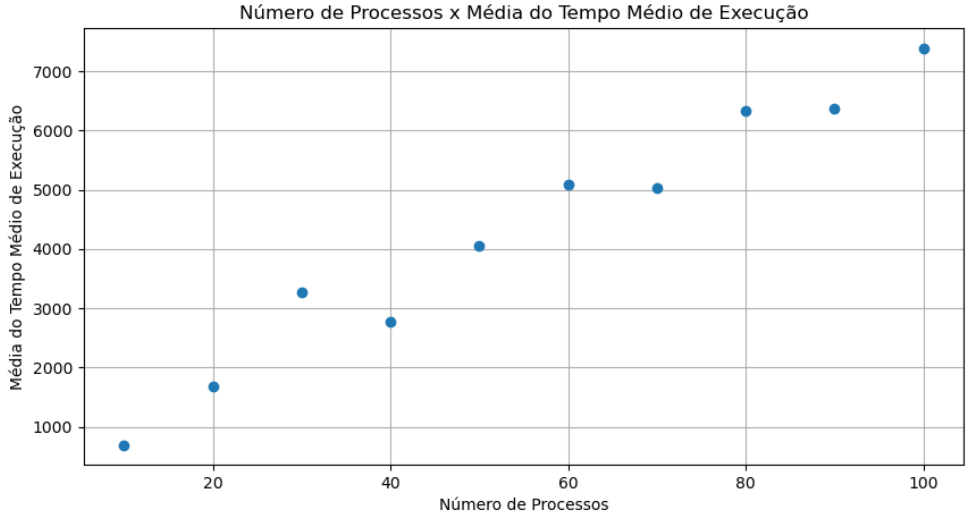


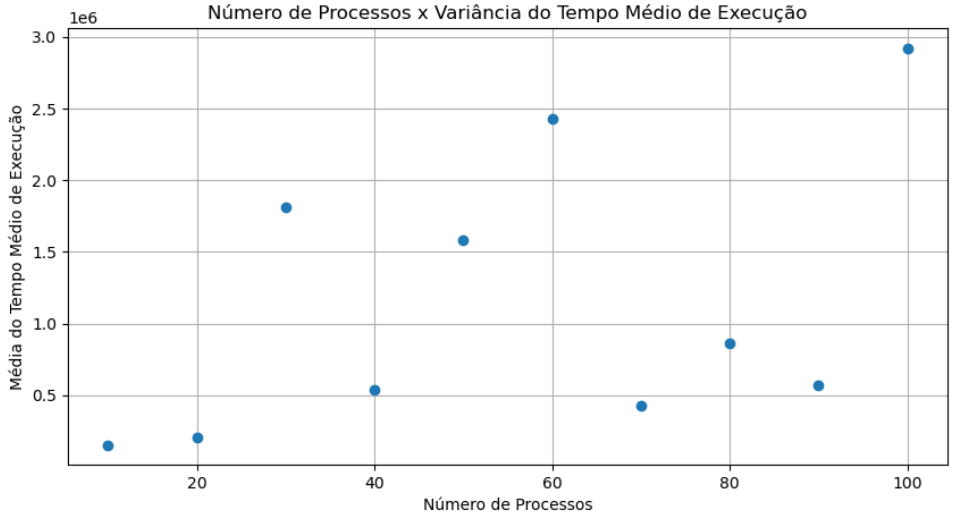
O algoritmo de prioridade estática (Prio_Static) apresentou desempenho variável conforme a distribuição das prioridades. Processos com prioridade alta tiveram tempos de espera baixos, enquanto aqueles com prioridade baixa experimentaram tempos de espera elevados. A variância do TME entre as amostras do Prio_Static foi moderada, refletindo a dependência da distribuição das prioridades no desempenho do algoritmo. Embora ofereça flexibilidade na definição de prioridades, esse algoritmo pode causar *starvation* para processos de baixa prioridade, resultando em um TME variável.





Por fim, a prioridade dinâmica (Prio_Dynamic) demonstrou ser a mais adaptável, ajustando o TME de acordo com o comportamento do sistema, o que resultou em uma distribuição mais equilibrada do tempo de espera entre os processos. A variância do TME para o Prio_Dynamic foi baixa, destacando sua capacidade de manter um desempenho consistente e equilibrado mesmo em condições de carga variáveis. Embora essa adaptabilidade seja uma vantagem significativa, a complexidade de implementação e a sobrecarga de processamento para ajustar as prioridades dinamicamente são desvantagens a serem consideradas.





Os resultados obtidos foram visualizados em dois gráficos principais. O primeiro gráfico apresenta os valores médios do TME para cada algoritmo. Ele evidencia a eficiência do SJF na redução do TME em comparação com os outros algoritmos, enquanto o FIFO e o LJF apresentaram TME mais elevados, refletindo suas desvantagens em determinados cenários.

O segundo gráfico ilustra a variância do TME entre as amostras de cada algoritmo. Essa análise de variância é crucial para entender a consistência do desempenho dos algoritmos sob diferentes condições. O SJF e o Prio_Dynamic mostraram menor variabilidade, indicando um comportamento mais previsível, enquanto o LJF apresentou uma maior variância, sinalizando que seu desempenho pode ser mais errático dependendo da carga de trabalho.

Conclusão

A análise comparativa dos algoritmos de escalonamento de processos revelou as seguintes conclusões:

1. **FIFO (First-In, First-Out)**: Simples e intuitivo, é eficaz para processos em ordem de chegada, mas pode resultar em altos tempos médios de espera para processos longos.
2. **SJF (Shortest Job First)**: Reduz o tempo médio de espera e resposta ao priorizar processos curtos, mas pode causar "fome" de processos longos.
3. **LJF (Longest Job First)**: Prioriza processos longos, o que pode ser útil em situações específicas, mas geralmente aumenta o tempo médio de espera.
4. **Prio_Static (Prioridade Estática)**: Eficiente para ambientes com prioridades estáveis, mas pode ser injusto com processos de baixa prioridade.
5. **Prio_Dynamic (Prioridade Dinâmica)**: Ajusta dinamicamente as prioridades, equilibrando melhor a carga e adaptando-se a necessidades variáveis, melhorando justiça e desempenho.

A escolha do algoritmo depende do contexto e das necessidades do sistema. O SJF é preferível para minimizar tempos de espera, enquanto o Prio_Dynamic é ideal para ambientes dinâmicos. Este estudo auxilia na compreensão dos trade-offs de cada algoritmo, informando decisões sobre o escalonamento de processos.

Referências

Lista de todas as fontes citadas no trabalho, seguindo o estilo de citação da SBC.

1. Aulas e arquivos de estudos (PDF) de autoria do Doutro João Batista Borges Neto
2. Páginas da internet como StackOverflow para maior compreensão da formação do algoritmo.