Relatório de Análise - Escalonador de Processos iCEVOS

Disciplina: Algoritmos e Estrutura de Dados I
Professor: Dimmy Magalhães

Integrantes:
José Francisco Paes Landim Sobrinho — Matrícula: 0030574
João Guilherme Aragão Malta — Matrícula: 0030617
Ricardo Cronemberger Cruz Ruben Pereira — Matrícula: 0030620

1. Justificativa de Design

A escolha da lista duplamente encadeada se baseou na sua eficiência para as operações que são cruciais para o escalonamento. A inserção de novos processos no final da fila é uma operação de tempo constante (O(1)), o que é fundamental para reinserir processos que ainda precisam ser executados. Da mesma forma, a remoção do primeiro processo da fila, que acontece quando ele é selecionado para rodar, também é O(1). Essa agilidade é perfeita para a lógica FIFO (First In, First Out) que usamos em cada nível de prioridade. Outra grande vantagem é a navegação bidirecional. Embora não seja usada de forma extensiva agora, ela nos dá flexibilidade para futuras otimizações e facilita a manutenção do código. Quando comparamos essa estrutura com outras, a superioridade fica ainda mais clara. Arrays e vetores, por exemplo, exigiriam um alto custo de deslocamento de elementos ou redimensionamento, algo que a lista duplamente encadeada evita completamente. Em relação a uma lista simplesmente encadeada, a versão dupla elimina a necessidade de percorrer toda a estrutura para inserir um novo processo no final. E, ao contrário de pilhas ou filas mais simples, ela nos oferece a flexibilidade que o escalonador necessita.

2. Análise de Complexidade

As operações mais importantes no nosso escalonador com lista duplamente encadeada são extremamente eficientes. A inserção no final e a remoção do início, como já mencionei, levam tempo constante, ou seja, O(1). A busca por um elemento específico é a única que leva tempo linear, O(n), já que exige uma varredura da lista. Já a verificação para saber se a lista está vazia é trivial, um simples teste O(1). Na prática, isso se traduz em um escalonador muito ágil. Cada ciclo de CPU, que envolve remover um processo, executá-lo, talvez reinseri-lo na fila e decrementar contadores, acontece em tempo constante. A mesma coisa se aplica ao desbloqueio de processos, à verificação do mecanismo anti inanição e à

migração entre as listas de prioridade. Por isso, a complexidade total de um ciclo é O(1), levando a uma complexidade de O(n) para n ciclos. A memória consumida é diretamente proporcional ao número de processos que estão no sistema.

3. Análise da Anti-Inanição

O mecanismo anti-inanição do iCEVOS funciona como uma regra de "equidade" para os processos. Ele é baseado em um contador que monitora os ciclos de alta prioridade. Quando esse contador alcança o número cinco, o sistema força a execução de processos de prioridade média ou baixa, e só depois reinicia a contagem. Nos demais casos, a preferência é sempre dada aos processos de maior prioridade que estiverem prontos. Essa abordagem garante que nenhum processo seja ignorado por tempo demais. Os processos de média prioridade, por exemplo, têm um tempo máximo de espera limitado a cinco ciclos, impedindo que sejam adiados para sempre. Já os de baixa prioridade, mesmo sendo executados com menos frequência, também têm a garantia de que serão atendidos. Sem essa medida, se houvesse um fluxo constante de processos de alta prioridade, os de níveis inferiores poderiam nunca ter a chance de rodar. Isso levaria a um acúmulo de tarefas, queda no desempenho do sistema e até problemas de estabilidade, já que esses processos acabariam ocupando recursos de forma improdutiva.

4. Análise do Bloqueio – Processo DISCO

A política de bloqueio pode ser vista de forma clara no ciclo de vida de um processo que precisa de acesso ao disco. Ele é criado e vai para a lista de prontos, esperando sua vez. Quando é executado e pede acesso ao disco, ele é movido imediatamente para a lista de bloqueados, sempre no final da fila, respeitando a lógica FIFO. Enquanto está bloqueado, ele não gasta ciclos da CPU, apenas espera o recurso ser liberado. A cada novo ciclo do escalonador, o processo que está há mais tempo na lista de bloqueados é liberado e volta para o final da sua lista de prioridade original. Na próxima vez que for executado, ele pode usar o recurso e, se não terminar, volta para o final da fila de sua prioridade. Essa política FIFO garante que todos os processos bloqueados sejam atendidos de forma justa, sem que um "fure a fila" do outro. Além disso, liberar apenas um processo por ciclo simula a limitação de recursos, mantendo o realismo e a coerência do nosso algoritmo de escalonamento.

5. Análise de Gargalos e Melhorias

O principal ponto de melhoria no sistema é a limitação de desbloqueio de recursos compartilhados. Atualmente, apenas um processo é desbloqueado por ciclo. Em cenários de alta demanda por acesso ao disco, isso se torna um gargalo, reduzindo

a eficiência do sistema, aumentando o tempo de espera e deixando a CPU ociosa. Para resolver isso, a proposta é usar um pool dinâmico de recursos. Em vez de liberar um por um, múltiplos processos poderiam usar o recurso simultaneamente, até um limite definido. Essa abordagem aumentaria o paralelismo e tornaria o sistema muito mais escalável, permitindo que o administrador ajuste a quantidade de recursos conforme a necessidade. A estimativa é que essa mudança poderia reduzir o tempo médio de bloqueio em até 80% e aumentar a utilização da CPU de 40% para 85%. Além disso, há outras melhorias possíveis, como priorizar processos de alta prioridade na fila de bloqueados, usar timeouts para liberar recursos que estão há muito tempo travados ou até mesmo implementar mecanismos de cache para reduzir a necessidade de acesso físico ao disco.