Análise Computacional do Simulador de Escalonador de Processos

Luis F. de Deus[1]
Centro de Tecnologia – Universidade Federal de Santa Maria (UFSM) 97.105-220 – Santa Maria – RS – Brasil
dedeus.f.l@gmail.com

Introdução

Nas duas primeiras execuções foram utilizadas as configurações de 50 unidades de tempo para a possível análise do *trace step by step*, com geração de processos a cada 1 unidade de tempo, o grau de multiprogramação do sistema era de 10 e a fatia de tempo do Round Robin foi de 5. Com estes parâmetros foram geradas duas execuções distintas para a análise do comportamento do simulador a uma resposta com muitos requerimentos de I/O, o chamado *I/O-Bound*, (Exec_1.txt) e para um cenário de poucas solicitações de I/O e alto desempenho de CPU, o chamado *CPU-Bound*, (Exec_2.txt).

Para uma terceira análise foi gerado uma execução com 200 unidades de tempo de simulação, grau de multiprogramação de 40 e fatia de tempo do Round Robin de 10, para a verificação de como o simulador se comportava com grandes volumes de dados, e a resposta das filas de espera do escalonador.

1. Primeira Execução

Para a Exec_1 foi utilizado intervalo entre solicitações de I/O de 2, e tempo de uso do dispositivo pelo processo de 6, forçando o escalonador a uma situação de muitas operações de I/O, como consequência houve a espera de processos na Lista do I/O para a obtenção do recurso, a qual está ordenada por prioridade, nesta configuração um processo que entrou na Lista cedo e com uma prioridade baixa, pode sofrer com a postergação indefinida (*Starvation*), onde outros processos podem preemptar sua execução.

Como o tempo entre solicitações de I/O é baixo (2) ocorrem muitas trocas de contexto na CPU, o processo em estado *Running* é sempre retirado para a alocação no estado *Bloqued*, a menos que seu *Burst Time* tenha acabado. Como a geração de processos

é a cada unidade de tempo o uso da CPU fica elevado, ficando ociosa somente no início da execução do simulador e nas trocas de contexto, bem como se há muitos processos a fila de chegada (*Arrive Queue*) tem um tamanho médio elevado devido a capacidade do sistema, ou seja, quando o sistema já comporta o número de processos do grau de multiprogramação, a *Arrive Queue* começa a aumentar de tamanho.

2. Segunda Execução

Para a Exec_2 foi utilizado intervalo entre solicitações de I/O de 6, e tempo de uso do dispositivo pelo processo de 2, assim o escalonador teria uma situação de pouco I/O e alta taxa de uso da CPU, porém como existem poucas solicitações de I/O e quando existem, o processo executa por pouco tempo, a lista do I/O fica ociosa, porque quando o processo requer o dispositivo o anterior a usá-lo já havia terminado, assim o processo que chega já obtém o recurso.

Outro fator importante é o comportamento da CPU, em função da frequência de solicitações de I/O os processos que obtém a CPU podem executar até a fatia do Round Robin terminar, ou seja, a menos trocas de contexto, logo o número de processos que terminam sua execução aumenta e consequentemente o uso da CPU que passa a ficar ociosa somente nos primeiros ciclos onde não há processo em *Ready* ainda.

3. Terceira Execução

Uma última execução foi feita, agora aumentando as características do sistema como, 200 unidades de tempo de simulação, grau de multiprogramação de 40, fatia de tempo do Round Robin de 10, geração de processos a cada 2 unidades de tempo, tempo entre solicitações de I/O de 5 e tempo de execução do I/O de 20, o intuito é analisar o simulador com resposta a uma grande execução de tempo e dados. Em resposta a isso é possível ver o aumento da capacidade de resposta do sistema como por exemplo a medida que as solicitações de I/O acontecem os processos são alocados na Lista, à medida que são criados vão para a fila de chegada, onde é verificado a capacidade do sistema que agora ficou maior então há um aumento de volume de processos, como consequência o simulador deve alocar cada processo em seu devido lugar.

Porém com o aumento do sistema é necessário notar os tempos de *turnaround*, logo aumentando o sistema quando o processo é realocado na fila de prontos para obter a

CPU, é o último, então vai ter que esperar um tempo maior até ganhar o recurso CPU novamente.

4. Comparação

Comparando as duas primeiras execuções se tem diferenças nas porcentagens de I/O e execução de CPU, onde na primeira a priorização é por uso de I/O, ou seja, uma execução de processos que requerem muitos acessos a disco ou periféricos como *keyboard* ou *mouse*, e na segunda execução a priorização é para a utilização da CPU com poucos I/O's simulando processos que requerem muito processamento lógico como por exemplo programas aplicativos, logo é possível notar que na Exec_1 em 94% do tempo algum processo estava usando I/O, isso faz com que o a eficiência caia bastante pois mesmo que a CPU não esteja ociosa, com 78% de tempo executando, aquele processo está bloqueado, pensando nele como um acesso ao disco por exemplo é um tempo em que o programa está parado esperando o dado do disco, esta característica é comum em sistemas computacionais com pouca memória RAM ou para usuários com muitas execuções "simultâneas" onde é requerido um determinado dado e o sistema não encontra na RAM, é então gerado um I/O para o disco.

Em contraponto a Exec_2 faz poucas solicitações de I/O, contabilizando 24% do tempo e o uso de CPU cresce bastante chegando a 96%, os tamanhos médios das filas enquanto na Exec_1 o tamanho médio da *Ready* era de 2, na Exec_2 é de 7, devido mais uma vez, a frequência de I/O pois quando se tem muitas solicitações o sistema fica com a maior parte dos processos em *Bloqued* na fila para usar o I/O enquanto na Exec_2 a fila de I/O fica sempre ociosa, logo os processos do sistema ficam a maioria em *Ready* esperando para ganhar o recurso CPU. Porém se for analisado o tempo de espera para ganhar a CPU, a Exec_1 é superior, leva menos tempo, devido ao fato das inúmeras trocas de contexto referente a processos que perdem a CPU para usar I/O, mas o tempo de *turnaround* da Exec_2 é maior, pois mais processos terminaram sua execução, o dobro da Exec_1.

A terceira execução aumentando os parâmetros de entrada, por consequência aumenta o tempo de *turnaround* médio, mas novamente mais processos terminam sua execução e o tempo está diretamente ligado ao tamanho da fila *Ready* por exemplo, pois quanto mais tempo demorar para ganhar a CPU, mais tempo vai demorar para finalizar sua execução, o tempo de espera para ganhar a CPU cresce devido ao tamanho do sistema

e a ordenação FIFO(*First In First Out*) da fila de prontos, as taxas de CPU e I/O são altas devido aos parâmetros utilizados para a simulação, 99% de CPU e 98% para I/O.