Algoritmos de Escalonamento na Comunicação entre Clientes e Servidor

Pedro Henrique Oliveira Sousa[1], Érica Carneiro Araújo[1]

[1]Ciências da Computação – Universidade Estadual do Ceará(UECE)
Ceará – CE – Brazil

pedro986@gmail.com, ericacarneiro.a@gmail.com

Abstract. *This report explains how it works, and the implementation decisions of five inter-thread scheduling algorithms for client/server communication, the language used was C++ and to work with threads the pthread library was used. The implemented algorithms were: FCFS (First-Come, First-Served), Lottery, Multiple Queue, Round-Robin and SJF (Shortest Job First). All these algorithms have one hundred and two threads, one is the scheduler, one is the server and one hundred are the clients.*

Resumo. *Este relatório explica como funciona, e as decisões de implementação de cinco algoritmos de escalonamento entre thread para a comunicação entre cliente/servidor, a linguagem usada foi C++ e para trabalhar com threads foi utilizada a biblioteca pthread. Os algoritmos implementados foram: FCFS (First-Come, First-Served), Loteria, Múltiplas filas, Round-Robin e SJF (Shortest Job First). Todos esses algoritmos possuem cento e duas threads, uma é o escalonador, uma é o servidor e cem são os clientes.*

1. Aspectos Gerais

Todos os algoritmos foram feitos em C++, para facilitar o uso de estruturas de dados. A biblioteca para a utilização das threads foi a pthread. Todos os algoritmos tiveram um código padrão, com as estruturas básica da comunicação cliente->servidor: feita por um vetor sendo o buffer de espera de dados a serem utilizados pelo servidor,o cliente só tem acesso a esse buffer quando é a vez dele de ser atendido, logo, esse buffer tem no máximo um elemento que deve ser processado pelo servidor, comunicação servidor ->cliente: no retorno da resposta para o cliente foi feito outro vetor, onde o valor randômico do cliente será o acesso para a comunicação neste sentido, antes de ser colocado algum valor, o valor desse vetor é zero, já que é uma resposta que não existe por parte do servidor em nenhuma circunstância. O servidor só irá parar quando ele souber que ele já atendeu o pedido de todos os clientes.

Para a verificação de tempo decorrido pelos algoritmos, no caso dos preemptivos, foi utilizada a biblioteca time.h do C, suas variáveis do tipo clock_t , para a contagem de tempo em relação ao clock da máquina, fazendo uma conversão para tempo em segundos para fazer a checagem se o tempo utilizado pela thread foi excedida.

2.Algoritmos

Os algoritmos escolhidos foram: FCFS(First-Come, First-Served), Loteria, Múltiplas filas, Round-Robin e SJF(Shortest Job First). Sendo os escolhidos para serem preemptivos o FCFS e Loteria.

Cada um possui 102 threads, 1 é o servidor, 100 são os clientes e 1 é o escalonador que vai decidir qual cliente irá ter acesso ao servidor.

A estrutura básica da thread servidor é: gerar todas as respostas possíveis para os clientes e em seguida aguardar a solicitação de uma thread cliente, acessando o vetor de comunicação cliente->servidor, conta quantos clientes já foram atendidos e então acessa o vetor de comunicação servidor->cliente com todos os dados já calculados pelo servidor e aguarda o pedido do próximo cliente.

A estrutura básica da thread cliente é: gerar o seu número randômico de 1 a 10000, então coloca seu valor no vetor de comunicação entre cliente->escalonador e salva o seu valor que servirá como seu identificador para ele saber quando será a sua vez de ser executado. Em seguida ele aguarda a decisão do escalonador e quando for a sua vez ele tem permissão de comunicar ao servidor qual é o seu valor e quando o servidor retornar o valor respondido o cliente agora irá saber o valor que ele estava esperando e pode finalizar a sua execução(no caso dos não preemptivos). Nos preemptivos foi adicionado um algoritmo que o cliente faz a fatoração do número retornado pelo servidor, dentro desse algoritmo ocorre a checagem do tempo que o cliente está em execução que é medida a partir do momento que o escalonador decidiu a sua vez e um ponto dentro desse algoritmo da fatoração.

A estrutura básica da thread escalonador é: A primeira coisa que todo escalonador possui é um aguardo para ele poder começar a executar apenas quando todos os clientes tiverem indicado no vetor de comunicação cliente->servidor os seus valores, para o escalonador decidir a ordem de execução baseada em todos os clientes. Em seguida é descrito o algoritmo de acordo com cada escalonador, a única semelhança de cada escalonador dentro das escolhas é que ele só escolherá o próximo cliente que poderá fazer a solicitação para o servidor quando o cliente anterior terminar a sua execução ou for interrompido (se o algoritmo for preemptivo).

Todos os algoritmos utilizaram mutex para a proteção das variáveis, garantindo a exclusão mútua dos algoritmos. Os algoritmos preemptivos tiveram um mutex a mais para garantir a proteção do aviso que a thread passou mais tempo do que ela deveria e deve ser chaveada para a vez da próxima thread, pelo escalonador. A main não é responsável pela inicialização e finalização dos mutexes, foi utilizado um construtor para inicializar todos os mutex, deixando a main apenas com a função de inicializar as threads.

Existe também a técnica de espera ocupada por parte de todos os algoritmos para definir momentos que a thread está aguardando resposta de outra thread para ela continuar a execução. Com essa técnica, o passo a passo do algoritmo irá sendo controlado através da espera ocupada, transformando a execução do código da seguinte forma: a função da main nos algoritmos é: criar as threads, iniciar a vez de quem deve

ser executado para nenhuma thread e setar a semente do rand, dessa forma pode-se controlar quem irá executar.

Primeiramente os clientes iram decidir os seus valores e o servidor irá calcular todas as respostas possíveis, quando todos os clientes tiverem dito quais são os seus valores no vetor de comunicação cliente->escalonador a vez de execução será passada para o escalonador através de uma variável que indica quem deve ser executado, no caso do escalonador foi decidido um número que não pertence a nenhuma thread cliente, sendo esse número escolhido muito alto, podendo ser considerado como infinito, em inteiros de 32 bits, sua representação hexadecimal é 0x3f3f3f3f.O escalonador irá começar a executar quando a vez for determinada por esse número, ele trabalhará em cima das informações passadas pelo vetor de comunicação cliente ->escalonador (valores_clientes) e poderá decidir quem irá executar, a partir desse momento todas as thread clientes estarão aguardando a sua vez de execução. Nesse momento existe o total controle de quem está em execução, pois primeiramente o escalonador decide qual cliente deve executar, em seguida o cliente fará a comunicação com o servidor através do vetor de comunicação cliente->servidor (buffer), apenas quando o servidor receber alguma informação ele irá executar e retornar o valor aguardado pelo cliente através do vetor de comunicação servidor->cliente (retorno), parando novamente a sua execução esperando a solicitação do próximo cliente ou finalizando sua execução, se já tiver atendido todos os clientes. O cliente então irá romper o aguarda de espera dessa informação e continuará a sua execução, printando o valor retornado, passando a vez para o escalonador e finalizando a sua execução, ou, no caso do código ser preemptivo, fará a fatoração do número retornado e printará as duas formas do número, o recebido pelo servidor e o calculado por ele.

Os preemptivos tiveram um acréscimo em seus algoritmos na thread cliente, para visualização da parada da thread por parte do escalonador, a atividade executada é a fatoração do número retornado pelo servidor. Dentro dessa atividade a thread cliente em execução é testada quanto tempo ele está em execução e se ela atingiu o tempo máximo. A forma utilizada para a contagem do tempo foi contar a partir do instante que a thread teve permissão do escalonador para acessar o vetor de comunicação cliente ->servidor, finalizando a contagem dentro do loop que está fazendo a fatoração do número retornado. A parada é determinada atrás de um teste de tempo que a thread já está em execução, se o tempo exceder o tempo determinado pelo código de 0.4 segundos, tempo escolhido por meramente ilustração por apresentar uma boa amostra de thread que são paradas e que seguem a sua execução sem pararem nenhuma vez. Caso o tempo seja excedido o código entra dentro de uma condição que irá comunicar a thread escalonador que a thread atingiu o tempo máximo permitido, a vez de execução passa a ser de nenhuma thread, pois enquanto não é a vez da thread escalonador ela está testando se esse sinal de interrupção chegou, se o sinal chegar a thread escalonador irá recolocar a thread que foi parada na fila de execução, o próprio valor da thread cliente é passado nessa comunicação para o escalonador saber quem parou.

2.1 FCFS - First-Come, First-Served

Esse algoritmo trata-se do primeiro que chega é o primeiro a ser servido, assim, de acordo com a ordem inserida dentro do vetor que contem as informações de todos os clientes, foi gerado uma fila. A ordem de execução desse escalonador será dado através dessa fila gerada, como esse é um dos algoritmos preemptivos, se uma thread cliente atingir o tempo máximo de uso da CPU, determinada pelo escalonador, ela será parada e colocada no final da fila. Quando a fila é finalizada,ou seja, todas as thread as 100 threads clientes foram atendidas, podendo a vez dela ter sido terminada mais de uma vez pelo escalonador, a thread escalonador é finalizada.

2.2 Loteria

Esse algoritmo ele escolhe aleatoriamente qual thread deve ser executada, distribuindo bilhetes para cada thread, tornado essa a sua chance de ser escolhida. Para essa implementação foi utilizado duas estruturas, vector(vetor dinâmico) e set(para determinar o conjunto de thread aguardando a execução).Para determinar quantos bilhetes teria cada thread foi tirado um mod quatro do seu número mais 1, dessa forma cada thread tem no mínimo um bilhete e no máximo quatro bilhetes para serem sorteados, colocando no final do vetor da loteria o valor da thread quantas vezes determinada pelo mod. Para escolher o bilhete foi tirado um randômico do tamanho do vetor total de bilhetes existentes, checasse até ser escolhido um número que ainda não terminou sua execução, quando acaba a execução de um cliente o seu valor é removido do set, assim o set vai reduzindo até terminar todos os clientes a serem executados, porém a chance dele ser escolhido sempre se mantêm intacto já que os valores não são removidos do vetor loteria. Como esse é o outro algoritmo preemptivo, a thread que foi interrompida tem seu valor retornado para o set, podendo ser escolhida novamente através do sorteio.

2.3 Múltiplas filas

Neste algoritmo as threads são divididas em algumas filas, no caso, foram divididas em quatro filas, a fila de prioridade três é a que tem maior prioridade para a execução entre as outras, esse algoritmo não é preemptivo, então após a inicialização de uma das threads a próxima só entrará em execução quando a que está em execução terminar. A forma de definir qual a prioridade de cada thread, foi tirar o mod quatro dos valores de cada thread, então a velocidade com que uma thread consegue sua vez para determinar o seu número, também influência na sua prioridade de execução dentro de cada fila determinada dentro do escalonador.

2.4 Round-Robin

A estrutura desse escalonador é a mais diferente em relação aos outros por si tratar de uma lista circular, por ter essa diferença, nesse algoritmo o vetor que recebe os valores de todos os clientes foi alterado para uma lista circular, com inicio no primeiro elemento adicionado na lista, por ser circula a estrutura foi modificada, passando a ser uma struct

de volatile int e um ponteiro para a próxima posição da lista circular. Com essa alteração foi necessário criar uma função que irá adicionar o novo valor dado na sua posição correta na lista, no caso, sempre no final da lista. Então as thread clientes não iram mais adicionar os valores dentro de um vetor, mas nessa estrutura. E o escalonador irá percorrer essa estrutura, executando todas as threads dentro da lista circular.

2.5 SJF - Shortest Job First

Nesse algoritmo o considerado mais curto será o primeiro a ser atendido, e assim sucessivamente, como não se sabe exatamente o tempo de processamento de cada thread com total certeza, foi decidido ordenar pelo valor do número do cliente, dessa forma, tende-se como tempo total de execução, o tempo que o servidor leva para descobrir qual é a resposta de cada número. Para fazer essa ordenação foi escolhida uma ordenação em O(N*logN), MergeSort. Então a atividade do escalonador é ordenar o vetor dos clientes com esse algoritmo e executá-los em ordem.

Referências

Tanenbaum, Andrew S. e Bos, Herbert (2016). Sistemas Operacionais Modernos, 4ª edição