PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

INF 1316
Relatório de Atividade - Lab 3

2210872 Felipe Antelo Machado de Oliveira
2112171 Pedro Henrique de Oliveira Valentim Gomes

Objetivo:

O objetivo do trabalho se trata de utilizar e comparar os conceitos de paralelismo e concorrência na manipulação de vetores suscetíveis a múltiplos acessos de uma vez. Dessa vez, teremos 2 exercícios. O primeiro trata de divisão eficiente de tarefas entre processos trabalhadores que operam em regiões distintas do vetor como a forma de exemplificação do dito paralelismo. Já no segundo experimento, todos os processos acessam simultaneamente as mesmas posições do vetor, permitindo observar os efeitos da concorrência. Ao final, é pedida uma análise dos resultados obtidos, indicando o motivo de tais ocorrências e as implicações práticas de cada uma das abordagens

Estrutura do Programa 1:

O programa 1 é estrutura seguindo a seguinte ordem: “Inicialização do Vetor”, “Criação dos Pipes”, “Criação dos Processos Filhos/Trabalhadores”, “Recebimento dos resultados (Processo Pai)”, “Período de espera pela Finalização dos Filhos” e, por fim, a “Saída Final”.

Para a “Inicialização do Vetor”, definimos um vetor principal a[10000] e inicializamos com o valor 10 em todas as posições, o que é feito através de um simples loop e inicialização de array.

Na “Criação de Pipes” são criados os 10 pipes para a comunicação de cada processo filho com o processo pai. Dessa vez a função específica pipe( ) é utilizada, além de um perror( ) para o tratamento de erros.

Na “Criação dos Processos Filhos/Trabalhadores” a dupla cria 10 processos com fork( ) com o intuito que cada processo execute de forma paralela uma parte do vetor e envie a soma parcial ao pai. Nesse caso, são utilizados o fork( ) para a criação do novo processo e o write( ) e close( ) para o envio e fechamento do pipe respectivo.

Na etapa de “Recebimento dos resultados (Processo Pai)”, o processo pai lê as somas parciais enviadas por cada um dos seus filhos e acumula no total. Funções utilizadas nesse processo são de abertura/leitura e fechamento de pipes, isso é, as funções read( ) e close( ).

No “Período de espera pela Finalização dos Filhos” é apenas uma etapa de

espera para garantir que todos os processos filhos terminem de enviar seus dados antes do fim do programa.

A “Saída Final” é a última das etapas, na qual é feita a impressão do resultado final da soma total de todos os processos.

Solução do programa 1:

Para a solução do programa 1 o grupo precisou aplicar e implementar o conceito de paralelismo por meio da criação de 10 processos filhos, onde cada um é responsável por processar uma parte distinta de um vetor de 10.000 posições previamente inicializado com o valor 10. Esse vetor é dividido igualmente entre os processos, com cada um atuando sobre 1.000 posições (10.000/10). Cada processo multiplica os valores da sua parte por 2 e calcula a soma parcial desses valores, que é então enviada ao processo pai através de um pipe. O processo pai, então, recebe as somas parciais de todos os filhos, soma os resultados em um valor final e imprime o valor final. Essa abordagem evita sobreposição de acesso ao vetor, e permite que os processos executem suas tarefas de forma independente e eficiente. Ao final, o valor total somado deve ser 200.000, resultado da multiplicação de todos os elementos do vetor (10 × 2) e da soma das 10.000 posições.

Estrutura do Programa 2:

O programa 2 possui uma estrutura relativamente similar àquela do programa 1, tendo uma divergência maior a partir do momento em que os processos filhos entram em ação. Dessa vez, cada processo filho atua **em todas as posições do vetor**, sem se preocupar com divisão de tarefas, simulando então uma concorrência não coordenada. Na leitura do processo pai (chamada anteriormente de “Recebimento dos resultados”), também há diferenças cruciais em que o processo pai recebe 10 vetores inteiros, um de cada processo filho, e os armazena em uma matriz (diferentemente do Programa 1, que recebe apenas inteiros com a soma parcial). A impressão também passa a ser de todos os valores dos vetores recebidos dos filhos, ao invés de somas parciais ou uma soma total. Aqui o foco maior é inconsistências e diferenças entre os vetores processados pelos filhos no exercício 1 e no exercício 2.

Solução do programa 2:

Para a solução do programa 2 a dupla precisou aplicar e implementar agora o conceito de concorrência. No exercício buscamos simular um ambiente concorrente de forma controlada, utilizando processos independentes que atuam simultaneamente sobre diferentes partes de um mesmo vetor. Inicialmente, é criado um vetor de 10.000 posições, todas preenchidas com o valor 10. Em seguida, o programa cria 10 processos filhos, onde cada um atua sobre 1.000 posições distintas do vetor, evitando sobreposição entre eles. Cada processo multiplica os valores de sua faixa por 2 e calcula a soma parcial do resultado. Essas somas parciais são enviadas ao processo pai por meio de pipes. O processo pai, atuando como coordenador, recebe as somas de cada trabalhador, acumula os valores e imprime o total ao final.

Observação e conclusões:

Ao todo, podemos afirmar que o trabalho foi um sucesso, pois, além da dupla

chegar no resultado desejado, nós pudemos também observar uma clara diferença entre os exercícios 1 e 2, devido a forma em que os dados dos vetores foram tratados. Esses 2 exercícios permitem observar na prática as diferenças entre paralelismo e concorrência.

No exercício 1(paralelismo), cada filho manipulando uma parte distinta do vetor, sem conflitos. Isso resultou em um programa eficiente, com divisão clara de tarefas e resultado previsível, evidenciando os benefícios de um paralelismo bem estruturado. Já no exercício 2, todos os processos executam as mesmas operações sobre o vetor inteiro, simulando um ambiente concorrente e competitivo. Isso levanta questões sobre os possíveis conflitos e sobre a redundância no processamento que mostra como a concorrência sem controle pode ser ineficiente e, em cenários com memória compartilhada, até fatal para o programa. Os resultados também se tornaram muito mais imprevisíveis quando comparados aos do exercício 1.

Assim, os exercícios destacam a importância de dividir corretamente as tarefas e, quando necessário, aplicar mecanismos de sincronização para garantir consistência e desempenho. Sendo assim, há uma clara diferença entre paralelismo e concorrência.