UNIVERSIDADE ESTADUAL DE MARINGÁ

GABRIEL DE MELO OSÓRIO
RA: 107862

MATHEUS AUGUSTO SCHIAVON PARISE
RA: 107115

Projeto Avaliativo

Programação Concorrente

Descrição do problema

O problema a ser resolvido consiste de uma série de desafios matemáticos contidos em vários arquivos a serem solucionados pelos computadores.

Há dois tipos principais de computadores, o primeiro é o cliente que é o responsável por iniciar todo o processo e será o computador utilizado em memória compartilhada. O segundo tipo é o servidor que será utilizado em memória distribuída, a função deste computador é esperar os comandos do computador cliente. Os arquivos serão transferidos pela rede para os computadores servidores, para o computador cliente ele irá receber a resposta dos servidores pela rede também. O computador cliente quando executar os cálculos de forma independente irá armazenar os resultados em arquivos diferentes. Os testes serão realizados de acordo com as quatro modalidades a seguir:

Memória Compartilhada - Sem Paralelismo

O computador cliente deverá executar os desafios por conta própria e de forma serial, isto é, um de cada vez sem nenhuma forma de paralelismo e sem auxílio externo.

Memória Compartilhada - Com Paralelismo

O computador cliente deverá executar os desafios de forma paralela com a utilização de *multithreading*, no caso realizado neste projeto não foi estabelecido um limite de memória RAM. Não haverá auxílio externo.

Memória Distribuída - Sem paralelismo

O computador cliente irá enviar os arquivos de problemas a serem resolvidos para os computadores servidores de forma aleatória e sequencial, estes resolveram os problemas de forma sequencial e irão devolvê-los ao computador cliente, que irá recebê-los sequencialmente. O computador cliente também pode ser usado como um computador servidor.

Memória Distribuída - Com paralelismo

O computador cliente irá enviar os arquivos de problemas a serem resolvidos para os computadores servidores de forma aleatoriamente e paralela, estes resolveram os problemas de forma paralela e irão devolvê-los ao computador cliente, que irá recebê-los simultaneamente. Assim como na modalidade acima o computador cliente pode ser usado como servidor.

Problemas Matemáticos

Os problemas matemáticos estão contidos em arquivos diferentes nomeados no formato “p” + N + “.txt”, onde N é o número do problema (exemplo p1.txt), o problema em sí não é o importante mas sim o tempo que cada um demora para ser resolvido e qual técnica é a mais adequada, isso também é influenciado pelo

número de arquivos dos problemas a serem resolvidos, há três problemas matemáticos:

- **Problema 1:** math.factorial(100000)
- **Problema 2:** math.factorial(1000000)
- **Problema 3:** math.factorial(2000000)

Eles são o fatorial de cem mil, um milhão e dois milhões respectivamente. Por escolha do grupo, o número de arquivos desse problema vai variar de 1 até 5, no final do processo os tempos vão ser somados para obter o tempo total.

Descrição da técnica

O modelo cliente-servidor é uma estrutura de aplicação que distribui as tarefas e cargas de trabalho entre os fornecedores de um recurso ou serviço. Neste trabalho a durante as modalidades de memória compartilhada o computador é tratado como sendo tanto cliente quanto servidor, pois o mesmo é o que solicita e resolve os problemas.

Na modalidade memória compartilhada sem paralelismo, o computador cliente (que também é servidor) lê cada um dos arquivos de forma sequencial, durante a leitura do arquivo ele analisa a equação escrita dentro do arquivo, faz o cálculo e em seguida cria um arquivo de resposta com formatação “p” + R + “.txt”, onde R é o número da resposta (exemplo r1.txt), ele cria um arquivo de resposta para cada arquivo de problema e em seguida finaliza a execução.

Na modalidade memória distribuída sem paralelismo o programa utiliza comunicação pela rede via Socket TCP, para isso ele necessita de endereços IP (ao menos um) e apenas uma porta para poder enviar a requisição, o computador servidor também precisará estar executando o programa de “servidor”, este estará configurado com o endereço IP do host e a porta escolhida para fazer a operação, neste caso o computador servidor irá executar de forma “sequencial”. O processo se inicia da seguinte maneira o computador cliente lê de forma sequencial cada um dos arquivos de problemas e para cada um deles ele cria uma conexão, envia o arquivo e depois de finalizar o envio do arquivo ele faz isso para o próximo até finalizar, após receber cada um dos arquivos os computadores servidores começam a processar as operações contidas neles e logo em seguida geram os arquivos de resposta. Por fim, os arquivos de resposta são enviados de volta para o computador cliente, um de cada vez.

Descrição da paralelização da técnica

Para a paralelização da técnica foi utilizado *multithreading*, isto é uma técnica que pode agilizar a velocidade de um programa caso o mesmo possua partes que possam ser paralelizadas, é como se um programa possuísse várias linhas de execução onde cada uma deles executa uma parte independente da outra, é

necessário tratar apropriadamente o espaço de memória e a porta do host que é utilizado por cada thread, .

Na modalidade memória compartilhada com paralelismo, o computador cliente (que também é servidor) lê o nome de cada um dos arquivos e cria uma thread para cada um deles, em seguida ele inicia todas as threads, dessa forma a leitura e análise de cada arquivo ocorre de forma paralela, ele faz o cálculo e em seguida cria um arquivo de resposta para cada um dos arquivos de problema simultaneamente e por fim finaliza a execução.

Na modalidade memória distribuída com paralelismo o programa também utiliza comunicação pela rede via Socket TCP, para isso ele necessita de endereços IP (ao menos um) e várias portas para poder enviar a requisição paralela de cada um dos problemas, o computador servidor também precisará estar executando o programa de "servidor", este estará configurado com o endereço IP do host e as portas escolhidas para fazerem as operações, neste caso o computador servidor irá executar de forma "paralela". O processo se inicia da seguinte maneira o computador cliente lê o nome de cada um dos arquivos e cria uma thread para cada um deles executar o envio dos arquivos de forma paralela, como o computador servidor está executando de forma paralela ele também irá receber cada um dos arquivos, processá-los e devolvê-los de forma paralela, isso tudo é possível pois cada um deles opera de forma independente, cada um em sua própria thread.

Tempo de execução de cada implementação

	Memória Compartilhada					
	Equação 1		**Equação 2**		**Equação 3**	
Paralelização	Não	Sim	Não	Sim	Não	Sim
1 Problema	0,1786	18,8037	9,8497	18,7074	31,4755	37,0948
2 Problemas	0,3636	18,4123	19,5258	18,7060	62,8591	37,3946
3 Problemas	0,5670	18,8346	29,4424	18,7233	94,3064	36,9143
4 Problemas	0,7540	18,8233	39,1704	18,7018	125,6292	37,6948
5 Problemas	0,9356	18,9572	48,9981	18,7118	156,7459	38,0948

	Memória Distribuída					
	Equação 1		**Equação 2**		**Equação 3**	
Paralelização	Não	Sim	Não	Sim	Não	Sim
1 Problema	0,2005	55,6818	10,5746	63,7246	33,7843	60,4273
2 Problemas	0,3983	69,2417	20,4842	80,9158	66,9181	59,9768
3 Problemas	0,5859	72,7981	30,3713	102,7981	99,5243	61,5546

4 Problemas	0,7664	79,3917	40,3222	117,3654	131,4857	60,2097
5 Problemas	0,9490	83,7981	50,2709	134,5149	165,4137	61,5423

Análise dos resultados

É possível inferir a partir destes resultados que só é de fato interessante paralelizar uma aplicação se o "problema" for grande o suficiente, para casos pequenos como o problema 1 (fatorial de cem mil) é mais interessante executar o programa de forma sequencial. O poder da paralelização se torna notável a partir do problema 2 (fatorial de um milhão) quando há pelo menos dois problemas sendo resolvidos ao mesmo tempo, quando estes são executados simultaneamente é como se o tempo para fazer em paralelo levasse o tempo de resolver apenas um problema quando ele está fazendo de forma sequencial, isto é considerando que o computador (seja ele cliente ou servidor) possua memória o suficiente para resolver estes problemas simultaneamente. Independentemente de utilizar memória compartilhada ou distribuída, essas afirmações são válidas para ambos os casos.

Ao comparar os resultados de memória distribuída com compartilhada, para esses problemas em específico é possível concluir que não é tão interessante usar memória distribuída. Ao executar os problemas com memória distribuída de forma serial eles tem um custo semelhante aos com memória compartilhada serial, não compensa ter o trabalho de transferir os arquivos pela rede. Ao utilizar paralelismo com memória distribuída, o custo de uso de threads somado ao tempo necessário para transferir o arquivo é muito maior que executar no próprio computador cliente.

Como conclusão para os casos específicos desse trabalho, a memória compartilhada com threads é melhor para resolver os problemas apresentados.

Link para o Google Drive o Projeto:
https://drive.google.com/drive/folders/1oCR8l_cy_qdvRNygAsgjgEMggUCjNsOi?usp=share_link

Bibliografia

Para o desenvolvimento deste trabalho foi utilizado o Python 3.8, com as bibliotecas *socket* para a conexão TCP, envio e recebimento de arquivos, *tqdm* para exibir as barras de progresso, *os* para manipular arquivos do sistema, *math* para fazer as operações com fatorial, *time* para contar o tempo decorrido, *threading* para o paralelismo com as *threads* e *PySimpleGUI* para interface do programa.