Programação Concorrente

Turma 01 de 2017.1
Aluno: Samuel Natã de França Borges
Matricula: 2014.027.959

Natal, RN. 20 de Abril de 2017

Introdução

Neste trabalho, foi realizado um experimento com o intuito de comparar a eficiência do uso ou não de threads com relação ao tempo de execução.

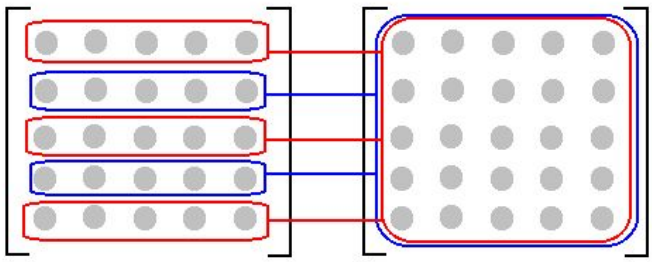
Experimento

O experimento consistiu em receber arquivos de entrada, cada arquivo contendo uma matriz, e obter o resultado da multiplicação dessas matrizes de duas formas, sendo uma com um algoritmo sequencial, e outra com um algoritmo concorrente fazendo uso de threads, por fim foi feita a análise dos tempos obtidos.
As matrizes são lidas da pasta **Matrizes** e possuem o nome *ANxM.txt* ou *BNxM.txt*, onde *N* é o número de linhas da matriz e M é o número de colunas. Todas as matrizes deste experimento eram quadradas, tendo *M=N*, e seguiam um padrão para o valor de N, com este valor indo de 2^i, com i variando de 2 a 11, totalizando 10 tamanho de matriz diferentes. O arquivo de uma matriz tem o seguinte formato:

N M
V[0,0] … V[0,M]
…. … ...
V[N,0] … V[N,M]

Quando o programa é rodado em modo automático, o tempo de leitura dos arquivos das duas matrizes a serem multiplicadas é salvo em um arquivo com o nome das duas matrizes, na pasta **readingFileTime**, já o arquivo **allTimes.dat** contém todos estes tempos.

Quando ao algoritmo sequencial, consiste de três laços encadeados (algoritmo de multiplicação de matrizes padrão), já o algoritmo concorrente é executado várias vezes, cada uma delas com um número *2^i* de threads, onde *i* varia de 1 a 11, e cada thread realiza multiplicações vetoriais, entre um vetor linha da matriz A com todos os vetores Coluna da matriz B, como mostra a figura abaixo, onde a cor vermelha representa a thread 1 e a os dados que ela irá utilizar em suas multiplicações, e a cor azul representa a thread 2, que funciona de maneira semelhante a 1.



Após a execução do algoritmo e obtenção dos resultados, foi salvo o tempo que se leva para imprimir a matriz resultado em, arquivo para o caso de rodar o programa no modo automático, e na tela para o caso de rodar o programa no modo manual (OBS: para alternar entre o modo manual ou automático, basta alterar o valor do macro AUTORUN no início do código e compilar novamente. **AUTORUN = true** roda no modo automático, e **false** roda no modo manual).

Resultados

O gráfico abaixo mostra o crescimento do número de elementos a serem lidos de arquivo e o crescimento do tempo gasto nesta tarefa. É importante lembrar que nesta parte do programa não foi feito uso de threads, logo não há distinção entre execução com e sem threads.

Observando o gráfico vemos que a quantidade de elementos a serem lidos cresce de forma proporcional ao tempo de leitura, o que faz sentido já que a leitura de um elemento possui tempo praticamente constante, logo conclui-se que o tempo de leitura de um arquivo é linearmente proporcional a seu tamanho.

A tabela abaixo mostra o tempo levado para encontrar a solução para o produto de duas matrizes de acordo com o tamanho ou grau das matrizes (N), e a quantidade de threads usadas nesta tarefa. Vale notar que quando o número de threads é 1, não se trata do uso de uma thread da biblioteca thread do **C++**, e sim do uso do algoritmo sequencial composto de 3 laços encadeados sem o uso da biblioteca thread.

Nº Threads	Grau da Matriz 4	8	16	32	64	128	256	512	1024	2048
1	1.94E-06	1.02E-05	7.04E-05	0.000562593	0.00189682	0.0154749	0.138162	1.31172	12.4915	154.185
2	0.000192908	8.71E-05	0.000113165	0.000681163	0.00165152	0.0140002	0.0834067	0.66633	6.31873	75.752
4	0.000187509	0.00011233	0.000126743	0.00127503	0.00123244	0.00653213	0.0442197	0.350517	3.7555	37.4548
8		0.000351468	0.000351943	0.000632985	0.00173728	0.00534513	0.0359965	0.191749	1.91824	19.8316
16			0.000919823	0.000453119	0.00230011	0.00454807	0.0333643	0.195214	1.93064	16.938
32				0.000819311	0.00190705	0.00430407	0.0252306	0.182786	1.90875	16.7207
64					0.00250459	0.00499259	0.02289	0.180795	1.88686	16.5095
128						0.00509623	0.0212352	0.175997	1.8708	16.5458
256							0.0222359	0.175707	1.85475	15.9548
512								0.180357	1.84541	14.9257
1024									1.8447	23.8951
2048										15.7313

Ao observar a tabela acima podemos concluir que de fato utilizar threads diminui o tempo de execução do algoritmo, porém podemos ver também que ao atingir um certo número de threads não há mais diminuição no tempo de execução do algoritmo. Uma análise melhor do ganho de velocidade pode ser dada pelo índice de aumento de velocidade **speedup**, que é dado pelo tempo da solução sequencial dividido pelo tempo da solução concorrente. Façamos a análise do **speedup** para o caso das matrizes de tamanho 2048 por 2048:

N° de Threads	Tempo	Speedup
2	75.752	2.035391805
4	37.4548	4.116561829
8	19.8316	7.774713084
16	16.938	9.102904711
32	16.7207	9.221204854
64	16.5095	9.339168358
128	16.5458	9.318679061
256	15.9548	9.663862913
512	14.9257	10.33016877
1024	23.8951	6.452578144
2048	15.7313	9.801160743

A partir da tabela acima é possível notar que o impacto do aumento do número de threads no ganho de tempo diminui conforme aumentamos este número, podendo chegar inclusive a diminuir o ganho de tempo. Vale ressaltar que o aumento do número de threads neste caso é quadrático.

Por fim, vejamos um gráfico do crescimento da quantidade de elementos da matriz resultado e do tempo de escrita da matriz resultado em arquivo:

Neste caso o gráfico ficou bem próximo ao do tempo de leitura, o que nós leva a crer que o tempo de escrita é linearmente proporcional ao quantidade de elementos, o que mais uma vez faz sentido já que o tempo de escrita é praticamente constante.

Conclusão

Ao fim deste experimento foi possível verificar que as threads podem de fato ajudar a diminuir o tempo de execução de tarefas computacionais, porém o uso inadequado de threads não só sobrecarrega o computador com o gerenciamento de uma quantidade absurda de threads, como também pode levar a um aumento no tempo de execução total da tarefa, sendo assim, para que não haja desperdício de recursos computacionais o uso de threads se faz indispensável, porem assim se faz também o seu uso com sabedoria, para que não ocorra novamente desperdício de recursos e tempo.