Primeiro Trabalho - Relatório Parcial

Universidade Federal do Rio de Janeiro

Disciplina: Computação Concorrente (MAB-117)

Período: 2021/1 Remoto

Título do Trabalho: Tabuadas de Multiplicação i x j

Nomes dos Membros: Gabriel Almeida Mendes, Luiz Rodrigo Lacé Rodrigues

DRE dos Membros: 117204959, 118049783

1 – Identificação/seleção e descrição do problema com paralelismo de dados

Escolhemos projetar um código em que basicamente calculamos várias tabuadas que são construídas usando como critério a passagem de dois parâmetros. Esses parâmetros são: o número de tabuadas (nTabuadas) e o número de múltiplos (nMultiplos). O número de tabuadas seria quantas tabelas seriam usadas percorrendo um valor de 1 até nTabuadas (i) e o número de múltiplos é até quantas operações o número atual da tabela é multiplicado percorrendo um valor de 1 até nMultiplos (j).

Ex: Para uma tabuada passando como argumentos:

nTabuadas = 3 e percorrendo de 1 até 3 (i)
nMultiplos = 14 e percorrendo de 1 até 14 (j)

Tabuada de 1 1 x 1 = 1 1 x 2 = 2 1 x 3 = 3 ... 1 x 13 = 13 1 x 14 = 14	Tabela de 2 2 x 1 = 2 2 x 2 = 4 2 x 3 = 6 ... 2 x 13 = 26 2 x 14 = 28	Tabuada de 3 3 x 1 = 3 3 x 2 = 6 3 x 3 = 9 ... 3 x 13 = 39 3 x 14 = 42

No final o objetivo é construir uma matriz que armazena as coordenadas cartesianas das tabelas. Usando o exemplo anterior ficaria:

i x j	**1**	**2**	**3**
1	1	2	3
2	2	4	6
3	3	6	9
...	...	...	...
13	13	26	39
14	14	28	42

O paralelismo entra quando podemos dividir a tarefa das threads, de multiplicar os números (i * j), em grupos/blocos em que cada uma fica responsável por um certo número de tabuadas. No final todas colocarão os resultados na matriz das coordenadas que será uma variável global compartilhada.

2. Projeto inicial da solução concorrente

A divisão das tarefas começa quando já temos o número total de resultados já pré determinados visto que podemos calculá-los por nTabuadas * nMultiplos, que resultaria na quantidade total de operações que seriam efetuadas, depois criamos uma matriz com essas dimensões. Em seguida, separamos as operações em blocos e cada um dos blocos ficará responsável por uma quantidade de tabuadas/tabelas a serem calculadas.

Essa estratégia foi escolhida visto que os valores são passados diretamente para uma variável global e não precisamos nos preocupar com um retorno nas threads.

O único argumento que precisamos passar para as threads é o id das threads, pois é com ele que fazemos o cálculo do tamanho dos blocos que serão criados e calculados. As outras variáveis serão todas de escopo global.

3. Projeto inicial dos casos de teste

Verificaremos a corretude comparando o resultado da versão concorrente com a da sequência e a verificação de desempenho será feito comparando tempo da tarefa que foi realizada de forma sequencial com a tarefa que foi realizada de forma concorrente.

Utilizamos a macro “timer.h” para fazer a tomada de tempo nos casos de desempenho da tarefa que estamos realizando, com um GET_TIME(ini) antes e GET_TIME(fim) depois que a tarefa foi realizada, onde o tempo de desempenho será dado pela diferença desses momentos (fim - ini).

Usaremos os seguintes casos, tentando-os no máximo 5 vezes cada, para fazer as avaliações:

Caso 1 - nTabuadas = 1000; nMultiplos = 100; nThreads = 1

Caso 2 - nTabuadas = 1000; nMultiplos = 100; nThreads = 2

Caso 3 - nTabuadas = 1000; nMultiplos = 100; nThreads = 4

Caso 4 - nTabuadas = 10000; nMultiplos = 500; nTheadas = 1

Caso 5 - nTabuadas = 10000; nMultiplos = 500; nTheadas = 2

Caso 6 - nTabuadas = 10000; nMultiplos = 500; nTheadas = 4

Caso 7 - nTabuadas = 500000; nMultiplos = 10000; nThreads = 1

Caso 8 - nTabuadas = 500000; nMultiplos = 10000; nThreads = 2

Caso 9 - nTabuadas = 500000; nMultiplos = 10000; nThreads = 4

4. Referências bibliográficas

Aulas gravadas da disciplina de Computação Concorrente.