Computação Concorrente, 2021.2 - Laboratório 3 - Ocorrência de Elementos em Array - Relatório

David Rodrigues Albuquerque - davidra@dcc.ufrj.br

Dezembro, 2021

Sumário

1 Resumo **2**

2 Configurações da Máquina **2**

3 Resultado Esperado **3**

4 Código **3**

5 Resultado obtido **4**

6 Conclusão **4**

1 Resumo

Este documento tem como por objetivo descrever os resultados obtidos e esperados do laboratório 3. Este laboratório consiste em projetar e executar um algoritmo capaz de realizar, dada uma entrada N = número de elementos do array, M = número de threads, L_i = limite inferior e L_s = limite superior, o cálculo do número de ocorrências de elementos quaisquer x tal que $L_i < x < L_s$, realizando o cálculo de forma sequencial e de forma concorrente com as M threads. Ao final, será feita a liberação da memória utilizada pelo programa.

O programa também deve ser capaz de avaliar o desempenho de todas as operações.

2 Configurações da Máquina

Para a execução do algoritmo do cálculo de ocorrências no array, foi utilizada as seguintes configurações da máquina:

- Arquitetura: x86-64
- CPU: Intel Core i7 10610U - 4 núcleos/8 threads - Frequência 2.3GHz até 4.9 GHz
- Armazenamento: SSD M.2 512 GB
- Memória: 16GB Ram DDR4 3200Mhz

3 Resultado Esperado

Serão utilizados diversos cenário de testes, sendo estes baseados no tamanho da quantidade de elementos no array. Esse array será preenchido com valores de ponto flutuante aleatórios.

- Tamanho 10^5: Será realizado o cálculo de forma sequencial e de forma concorrente utilizando 1/2/4 threads
- Tamanho 10^6: Será realizado o cálculo de forma sequencial e de forma concorrente utilizando 1/2/4 threads
- Tamanho 10^7: Será realizado o cálculo de forma sequencial e de forma concorrente utilizando 1/2/4 threads
- Tamanho 10^8: Será realizado o cálculo de forma sequencial e de forma concorrente utilizando 1/2/4 threads
- Tamanho 10^9: Será realizado o cálculo de forma sequencial e de forma concorrente utilizando 1/2/4 threads

Cada um desses cenários de testes será executado 5 vezes, de forma a minimizar interferências geradas pelo sistema operacional na velocidade e tirar o tempo de execução mais baixo.

Espera-se que, em casos menores, o tempo de execução do cálculo sequencial seja mais eficiente ou igual, por ser um cálculo pequeno. Desta forma, o cálculo sequencial não desperdiça tempo escalonando as threads. Conforme cresce-se o tamanho da quantidade de elementos no array, devemos perceber um desempenho superior na execução concorrente.

4 Código

O código utilizado na execução do programa pode ser obtido no seguinte repositório.

https://github.com/Albuquerque-David/concurrent-computing/tree/main/lab3

5 Resultado obtido

Mínimo encontrado em todos os casos.
Tabela gerada através do result.py.

Tipo	Tamanho: 10^5	Tamanho: 10^6	Tamanho: 10^7	Tamanho: 10^8	Tamanho: 10^9
Sequencial	0.000683	0.007004	0.069311	0.697615	6.977347
Concorrente: 1 Thread	0.000851	0.007222	0.06978	0.692759	6.944598
Desempenho: 1 Thread	0.8025851938895417	0.9698144558294101	0.9932788764689023	1.0070096527075072	1.0047157517252978
Concorrente: 2 Thread	0.000516	0.00377	0.035313	0.349507	3.498885
Desempenho: 2 Threads	1.3236434108527133	1.8578249336870027	1.962761589216436	1.9959972189398207	1.994162997640677
Concorrente: 4 Thread	0.000405	0.002212	0.017935	0.175112	1.811347
Desempenho: 4 Threads	1.6864197530864198	3.1663652802893307	3.86456649010315	3.983821782630545	3.852021175401 5103

6 Conclusão

Como esperado, podemos observar que, para valores pequenos do array, o cálculo sequencial é equivalente ao desempenho do cálculo realizado de forma concorrente ou levemente superior, como no caso do cálculo sequencial do array $N = 10^5$, $N = 10^6$ e $N = 10^7$. Mas, assim que escalamos o tamanho do array, podemos observar mudanças significativas e proporcionais ao tamanho das threads. Quando se utiliza 2 threads, o desempenho tende a dobrar, e quando se usa 4 threads, o desempenho tende a quadruplicar.

Utilizando a Lei de Amdahl nas linhas do desempenho para o cálculo, podemos observar que de fato o paralelismo está sendo realizado, nos fornecendo esse ganho de desempenho.