Avaliação de desempenho

Exercício 2 Atualizado

2024.1

Alunos: Camila Xavier (CXM) & Vituriano Xisto (VOX)

Observação: refizemos o exercício 2 para conseguirmos um melhor desempenho!

Mudanças:

No código novo, cada célula da matriz resultante é calculada por uma goroutine separada (multiplyRowColumn), permitindo um alto grau de paralelismo. Cada goroutine é responsável por calcular o valor de uma célula da matriz resultante, distribuindo assim o trabalho de forma mais equilibrada e eficiente para grandes matrizes.
No código antigo, a função MultiplicaMatrizesSync paraleliza apenas o cálculo das linhas da matriz resultante, mas cada linha é calculada inteiramente dentro de uma única goroutine, reduzindo o nível de paralelismo e resultando em menor eficiência para grandes matrizes.

Além disso: No código novo, o uso de mutex é limitado ao ponto em que cada célula da matriz é escrita, minimizando a contenção. Cada goroutine calcula uma célula independentemente e só usa o mutex para atualizar a célula resultante.
No código antigo, o mutex é usado dentro dos loops internos para cada multiplicação de elementos da linha e coluna, resultando em alta contenção e redução de desempenho, especialmente quando muitos goroutines tentam acessar o mutex simultaneamente.

Por fim, o novo código usa o WaitGroup eficientemente para garantir que todas as goroutines terminem antes de retornar a matriz resultante, enquanto
o código antigo também usa WaitGroup, mas a paralelização em um nível mais grosseiro (por linha) resulta em menor eficiência.

Novos valores do exercício 2:

1) **Objetivo:**

Realizar uma avaliação comparativa de desempenho das duas versões implementadas no Exercício 01: uma versão sem concorrência e uma versão concorrente usando mutex. Considerar matrizes de tamanhos distintos: 10x10, 100x100 e 1000x1000 (rodar 30x).

2) **Listar os serviços do sistema:**

Multiplicação de matrizes utilizando:

- Método sequencial (sem concorrência)
- Método concorrente (usando mutex)

3) **Escolher as métricas de desempenho:**

Medição de tempo de execução da aplicação (multiplicação de matrizes).

4) **Listar parâmetros:**

Parâmetro do Sistema	**Valor**
Hardware	MacBook M2 Pro, 16GB
Sistema Operacional	macOS (Sonoma 14.4.1)
Linguagem de programação	GO
Fonte de alimentação	Rede elétrica
Processos em execução	Apenas os estritamente necessários à realização do experimento
Interfaces de rede	Desligadas

5) Escolher fatores:

Fator	Nível
Mecanismo de sincronização	Mutex, sem concorrência
Tamanho da Matriz	10x10, 100x100, 1000x1000

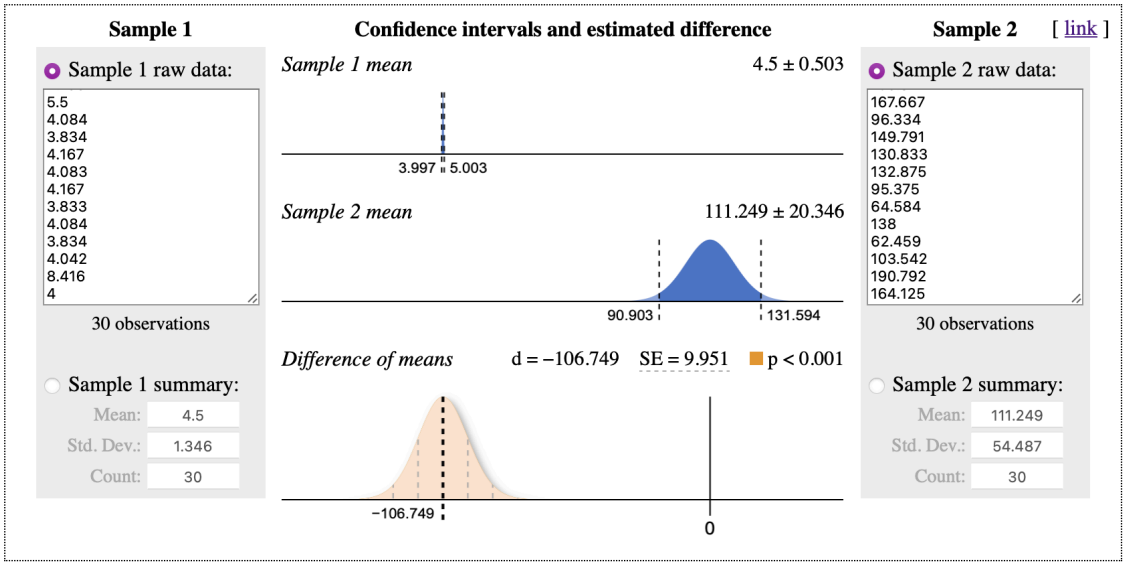
6) Escolha a técnica de avaliação

Execução repetida de experimentos para cada combinação de parâmetros e fatores, medindo o tempo de execução.

7) Resultados:

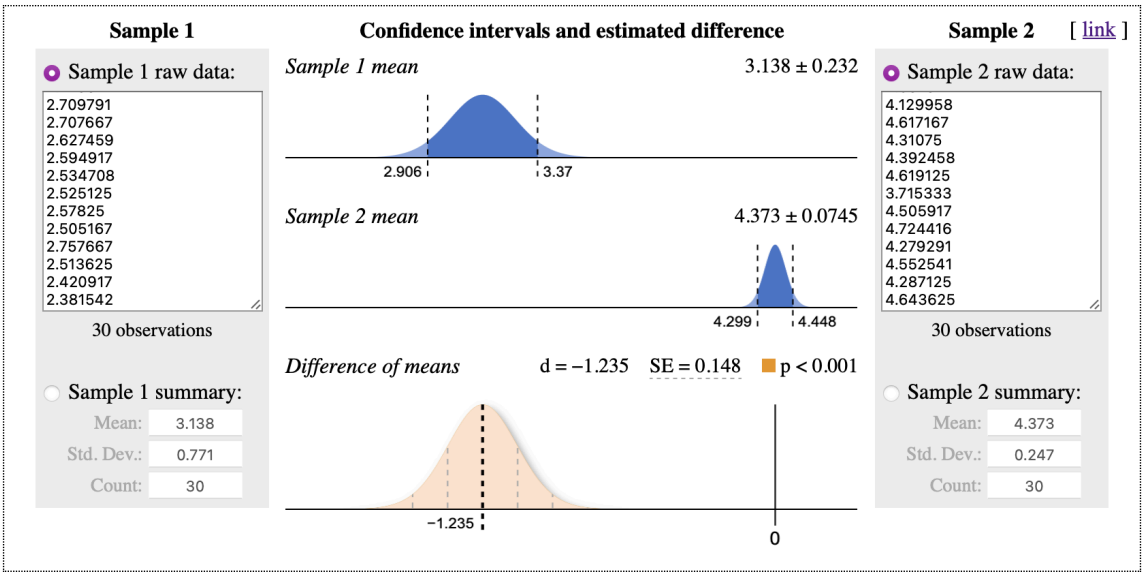
Sample 1 : sem concorrência
Sample 2 : com concorrência

10x10



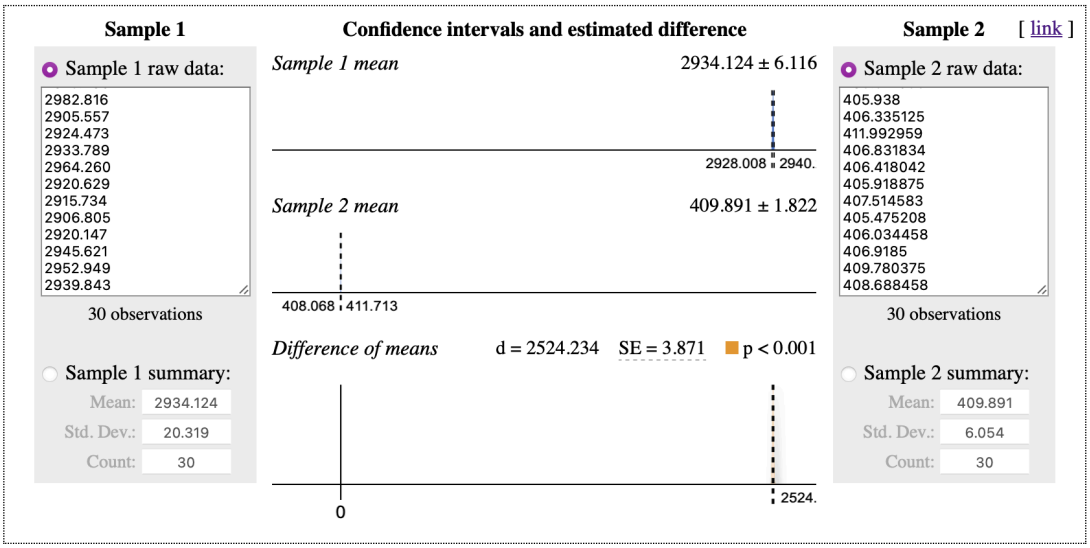
Verdict: Sample 2 mean is greater

100x100



Verdict: Sample 2 mean is greater

1000x1000



Verdict: Sample 1 mean is greater

8) Análise dos resultados:

Sobrecarga de Concorrência:

Para matrizes pequenas (10x10 e 100x100), a sobrecarga de gerenciamento de concorrência (criação e sincronização de *goroutines*) supera os benefícios, resultando em pior desempenho comparado à execução sequencial.

Benefícios de Concorrência para Matrizes Grandes:

Para matrizes grandes (1000x1000), a concorrência mostra um desempenho significativamente melhor, reduzindo drasticamente o tempo de execução. Isso indica que a paralelização é vantajosa quando a carga de trabalho é suficientemente grande para amortizar a sobrecarga de gerenciamento de *goroutines*.

Estabilidade do Desempenho:

A execução sequencial é mais estável e previsível em termos de tempo de execução.

A execução concorrente tem maior variação em tempos de execução para matrizes pequenas, mas é eficiente para grandes tamanhos.

Esses resultados destacam a importância de considerar o tamanho da tarefa ao decidir entre execução concorrente e sequencial. Concorrência pode ser altamente benéfica para tarefas de grande escala, mas pode ser contraproducente para tarefas menores devido à sobrecarga associada.