Introdução

Usar o Partitioner permitiu dividir os blocos de processamento evitando o overhead quebrar o processamento em blocos muitos pequenos onde o custo-benefício do paralelismo pode inviabilizar a solução

Em relação ao uso do SIMD, mostra como a otimização via hardware ganha em performance em todos os cenários. Fazer operações com SIMD exije menos instruções necessárias para processar os mesmos dados. Dessa forma, reduz acessos à memória e movimentação de dados entre registradores e memória principal.

Experimento

```
Microsoft Visual Studio Debu X
  --> Reading from .\images
  --> Writing into C:\Users\lucas\AppData\Local\Temp\ImageGalleryOutput
This is the Image Processing Example
This program converts a bunch of images to grayscale
(1) Running in a Single Thread ...
Elapsed time: 55133 ms
(2) Running in Multi Threads -- static partitioner ...
Elapsed time: 15458 ms
(3) Running in Multi Threads -- dynamic partitioner ...
Elapsed time: 11575 ms
(4) Running in Multi Tasks -- static partitioner ...
Elapsed time: 18985 ms
(5) Running in Multi Tasks -- dynamic partitioner ...
Elapsed time: 24773 ms
(6) Running with SIMD Tasks...
Elapsed time: 469 ms
C:\Projects\INF0082\[M4][Lab02]ImageGalleryLab\bin\Release\net8.0\ImageGalleryLab.exe
Press any key to close this window . . .
```

Usando a execução em modo Release, foram obtidos os resultados acima. A solução usando SIMD se confirmou melhor, como já era esperado, mas outros resultados interessantes foram gerados:

- A solução usando Threads funcionou melhor com o Partitioner, no geral, pela característica do Parallel usar melhor os benefícios do Thread Pool do .NET;
- Por outro lado a solução com Tasks acaba causando overhead por criar várias tasks assíncronas;

Importante notar que o profile mostrar maior tempo de CPU por parte da solução single thread, seguido das soluções usando thread:

```
CPU Usage - ...3.diagsession + X Report202501...3.diagsession*
                              Console.WriteLine("(1) Running in a Single Thread ...");
                              var sw = Stopwatch.StartNew(); // Start the stopwatch
   45138 (16,96%)
                               GrayScale.ProcessST(imageFiles, Path.Combine(outputDir, "SingleThread"));
                               sw.Stop(); // Stop the stopwar
                               Console.WriteLine($"Elapsed time: {sw.ElapsedMilliseconds} ms");
        3 (0,00%)
                               Console.WriteLine("(2) Running in Multi Threads -- static partitioner ... ");
                               sw.Restart(); // Restart the
                               GrayScale.ProcessMT(imageFiles, Path.Combine(outputDir, "MultiThread_StaticPart"), true);
     9283 (3,49%)
                               sw.Stop(); // Stop the stopwatch
                               Console.WriteLine($"Elapsed time: {sw.ElapsedMilliseconds} ms");
                       29
                               Console.WriteLine("(3) Running in Multi Threads -- dynamic partitioner ... ");
                               sw.Restart(); // Restart the
                               GrayScale.ProcessMT(imageFiles, Path.Combine(outputDir, "MultiThread_DynPart"), false);
     8323 (3,13%)
                               sw.Stop(); // Stop the stopwat
                               Console.WriteLine($"Elapsed time: {sw.ElapsedMilliseconds} ms");
                               Console.WriteLine("(4) Running in Multi Tasks -- static partitioner ...");
                               await GrayScale.ProcessTasks(imageFiles, Path.Combine(outputDir, "MultiTask_StaticPart"), true);
        5 (0.00%)
                               sw.Stop(); // Stop the stopwatch
                               Console.WriteLine($"Elapsed time: {sw.ElapsedMilliseconds} ms");
                               Console.WriteLine("(5) Running in Multi Tasks -- dynamic partitioner ...");
                               await GrayScale.ProcessTasks(imageFiles, Path.Combine(outputDir, "MultiTask_DynPart"), false);
        1 (0.00%)
                               sw.Stop(); // Stop the stopwato
                               Console.WriteLine($"Elapsed time: {sw.ElapsedMilliseconds} ms");
                               Console.WriteLine("(6) Running with SIMD Tasks...");
                              sw.Restart(); // Restart the stop
                               await GrayScaleSIMD.ProcessTasks(imageFiles, Path.Combine(outputDir, "SIMD"));
        5 (0,00%)
                               sw.Stop(); // Stop the stopwatch
                               Console.WriteLine($"Elapsed time: {sw.ElapsedMilliseconds} ms");
                           catch (Exception ex)
91 %
           No issues found
Dutput
```

A granularidade do processamento e troca de contexto podem ser as razões principais do consumo de maior tempo de CPU.

Conclusão

Os experimentos mostram como diferentes abordagens de paralelismo e otimizações de hardware podem impactar o desempenho de uma solução. O uso do Partitioner mostrou ser eficaz ao dividir o trabalho em blocos adequados, evitando o overhead que ocorre quando o processamento é fragmentado em blocos excessivamente pequenos.

Por outro lado, o uso do SIMD comprovou seu potencial de ganho de desempenho, destacando-se ao reduzir o número de instruções necessárias para o processamento de dados, além de minimizar o custo de movimentação de dados entre registradores e memória.