Universidade do Minho
Escola de Engenharia

Exercício Nº6 - Transformada da Distância

Computação Paralela
(4º/1º ano MIEI/MEI)

Daniel Costa
(a81302)

João Marques
(a81826)

14 de Dezembro de 2019

Conteúdo

Capítulo 1

Introdução

Este relatório mostra e compara o desempenho de diferentes implementações do exercício **Nº 6** da Unidade Curricular de *Computação Paralela* denominado ***Transformada da Distância***. O exercício consiste em transformar uma imagem *PBM* de apenas duas cores (branco e preto) numa escala de tons de cinzento desde a cor preta até à cor branca.

Para calcular a imagem resultante são efetuadas sucessivas iterações sobre a mesma até esta não conter mais nenhum pixel branco. A forma como é calculada a cor de um pixel é observando o valor de todos os seus vizinhos, guardando o menor valor destes e somando-se 1 ao menor, criando assim uma escala de tons de cinzento.

O objetivo é criar versões sequenciais e paralelas deste problema de forma a discutir quais são os ganhos possíveis retirar das implementações paralelas aplicadas às implementações sequenciais, sendo mostrado no próximo capítulo as implementações sequenciais, seguidas das implementações paralelas e por fim a análise de resultados obtidos.

Capítulo 2

Implementação Sequencial

2.1 Simplista

A implementação sequencial recebe uma matriz e calcula a distância de cada ponto ao ponto de fundo mais próximo. Assim, percorre cada ponto da matriz, excetuando as bordas da matriz, e calcula o mínimo dos pontos circundantes de cada ponto. Posteriormente, guarda o resultado numa matriz auxiliar o sucessor natural do mínimo calculado. Esta implementação executa iterativamente este processo até não haver mais pontos por processar. O compilador otimizou esta implementação para usar *loop unrolling* nos dois loops interiores.

2.2 Loop Tiling

A implementação usando *tiling* tem como objetivo aproveitar a localidade temporal e os benefícios da cache. Esta implementação é similar à implementação simplista mas divide a imagem em *tiles* que teoricamente cabem na cache. No entanto esta versão tem maior *overhead* por iteração porque acresce dois *loops* e menor localidade espacial.

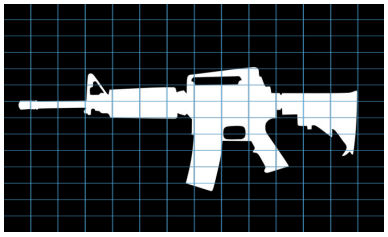


Figura 2.1: Divisão em Tiles de uma imagem

2.3 Loop Tiling otimizado

Esta versão do algoritmo é semelhante à versão **Loop Tiling** mas tenta poupar trabalho em partes da imagem já processadas. Nesta versão tem uma estrutura que guarda os *tiles* que ainda não estão totalmente processados. Isto permite que por cada iteração se reduza o número de *tiles* processados.

Capítulo 3

Implementação Paralela

3.1 Simplista Parallel For

A solução mais simples para tornar o código paralelo é aplicar a diretiva *omp parallel for* ao *loop* que percorria cada linha da imagem. Assim, dividimos a carga de cada tarefa por linhas. A este loop aplicamos diferentes escalonamentos, nomeadamente *static*, *guided* e *dynamic*. Empiricamente, as imagens não costumam ser uniformes e, por isso, diferentes regiões da imagem terão cargas diferentes. Em teoria, os métodos *guided* e *dynamic* devem balancear a carga de trabalhos e por esse motivo obter melhores resultados do que o *static*.

3.2 Simplista Parallel For Collapse(2)

Posteriormente, experimentamos dividir a carga de cada tarefa por linhas e colunas. Teoricamente, esta implementação irá melhorar a performance de imagens com reduzido número de linhas. Por outro lado, pode

ajudar a balancear a carga de cada *thread*. A este *loop* aplicamos diferentes escalonamentos, nomeadamente *guided* e *dynamic*.

3.3 Parallel Tiling

Aproveitando os benefícios da versão sequencial com *tiling* aplicamos a diretiva *omp parallel for* ao *loop* que percorre cada um dos *tiles*. Este versão procura otimizar a utilização da cache. O tamanho do *tile* é dimensionado para caber em metade da cache, permitindo ter parte das duas matrizes em cache. Teoricamente, na iteração seguinte o valor escrito na iteração anterior pode já estar em cache permitindo assim acelerar os acessos à memória. No entanto, quando o número de *threads* é superior ao número de *cores* físicos espera-se reduzir o desempenho porque ambas as *threads* vão realizar as mesmas instruções e reduzir a cache disponível.

3.4 Parallel Tiling otimizado

Por fim, paralelizamos a versão sequencial **Tiling otimizado** aplicando a diretiva *omp parallel for* ao *loop* que percorre cada um dos *tiles*. Esta versão terá as vantagens da versão otimizada sequencial mas, em princípio, não terá grande grande *speed up* em relação à versão sequencial otimizada. Isto acontecerá porque, em princípio, nas últimas iterações o balanceamento não será ideal. Devido ao balanceamento ser ainda mais desigual do que a versão **simplista** paralela o escalonamento *static* seria terrível. Teoricamente, o escalonamento *guided* e *dynamic* devem obter bons resultados e destribuir o trabalho entre as diferentes *threads*.

Capítulo 4

Análise de Resultados

Para análise de resultados foram realizados testes às versões sequenciais e às versões paralelas com três imagens diferentes (Gun 8000x4500, Finger Print 1592x2312 e Floppy Disk 16x16), os tempos apresentados para cada uma das versões e para cada uma das imagens é a mediana das 8 execuções.

De seguida são ilustrados os resultados das versões paralelas com 1(idênticas ao sequencial), 2, 4, 8, 10, 16, 20, 32 e 40 *threads*, para cada uma das imagens realizadas. Para um melhor controlo e melhores resultados foram reservadas todas as *threads* lógicas do nodo do SeARCH que estava a ser utilizado para a realização dos testes, para garantir que não existiam mais programas a gerar concorrência de forma a adulterar resultados.

Nas seguintes imagens podemos visualizar as curvas de ganho da imagem Gun e os tempos de execução das versões sequenciais para a mesma. As curvas de ganhos e tempos de execução das outras imagens encontram-se nos anexos do relatório.

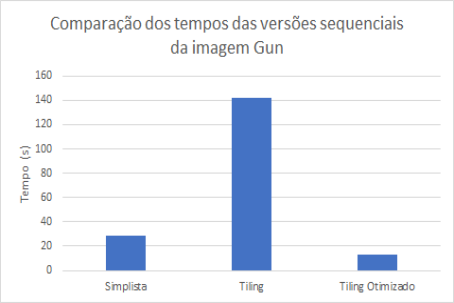


Figura 4.1: Desempenho das versões sequênciais

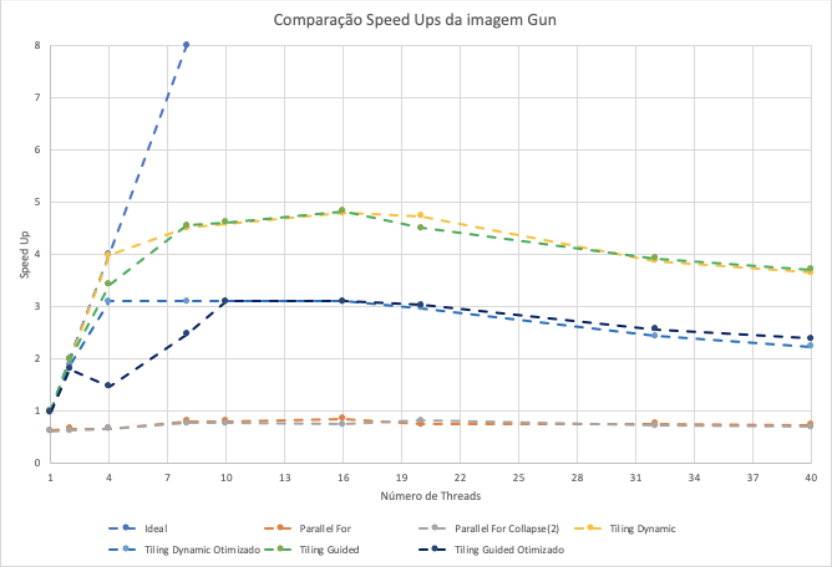


Figura 4.2: Curvas de ganho da imagem Gun

4.1 Sequencial

Os tempos das versões sequenciais mostraram piores desempenho do **Tiling** e do **Tiling Otimizado** do que a versão **Simplista**. Os resultados mostram que quanto maior for a imagem maior é a disparidade entre os tempos do **Tiling** e **Simplista**. Por outro lado, o **Tiling Otimizado** conseguiu ter melhores tempos do que a versão **Tiling**. Para imagens pouco uniformes, como é o caso da **Gun**, conseguiu-se ter melhores resultados do que a implementação **Simplista**. O **Tiling**, apesar de ter maior localidade temporal, tem mais saltos a posições não contínuas de memória o que provoca maiores acessos à memória. A versão **Tiling Otimizado** apesar de ter maior tempo de montagem e mais operações por ciclo, demonstrou que compensava pois reduzia ser processados grandes porções da imagem.

4.2 Paralelo Floppy Disk

Para imagens de pequena dimensão, tal como a imagem Floppy Disk, não existem ganhos de tempo de execução, devido à sincronização entre threads e divisão de trabalho, as implementações paralelas são substancialmente mais lentas que a implementação sequencial, tal pode ser visualizado nas imagens dos

gráficos em anexo e respetiva tabela de tempos de execução e Speed Up.

4.3 Paralelo Finger Print

No caso de imagens de tamanho médio, como a imagem Finger Print, foram obtidos alguns ganhos com as implementações **parallel for** e o **parallel for collapse(2)**, a implementação parallel for collapse(2) foi realizada para testar se existiam alguns ganhos na paralelização dos dois loops em vez de apenas um, mas pelos dados em anexo podemos verificar que não, relativamente à implementação sequencial **simplista**. Com as implementações de **tiling** obtiveram-se maiores ganhos apesar de estas versões serem mais lentas, pois a versão sequencial de **tiling** é mais lenta que a versão sequencial **simplista**. Quando utilizado o **tiling otimizado** são obtidos alguns ganhos temporais, apesar do Speed Up ser idêntico à versão com **tiling**), mas apenas quando são utilizadas poucas threads. Como os *tiles* tem dimensão 256x2048 a imagem é dividida em apenas 14 *tiles* pelo que quando são utilizadas mais threads que o número de *tiles* deixa de haver melhoria do Speed Up e melhoria de tempos de execução.

4.4 Paralelo Gun

Para imagens de grande dimensão como a imagem Gun, as versões paralelas de **parallel for** e **parallel for collapse(2)** não obtiveram qualquer ganho em relação à versão sequencial **simplista**, isto deve-se principalmente aos acessos à memória, pois como as linhas são muito grandes e no cálculo de uma célula são necessários acessos à linha anterior e à linha seguinte, resultando tem muitos cache miss. Para as versões com **tiling** e **tiling otimizado** foram observados ganhos substanciais no *speed up*, principalmente na versão não otimizada. No entanto, o tempo de execução mais rápido foi observado na versão otimizada. Não se obtiveram mais ganhos devido ao balanceamento de carga (nas últimas iterações só algumas *threads* tem trabalho para realizar e também a sincronização, pois no final de cada iteração do processamento da imagem é necessário esperar por todas as *threads* antes de iniciar a nova iteração.

Capítulo 5

Conclusão

Em suma, paralelizar a versão **Simplista** não melhora o desempenho. Os resultados mostraram que **Tiling** apenas consegue bons resultados quando se utiliza várias *threads*, principalmente com escalonamento dinâmico, e conclui-se que o **Tiling Otimizado** com escalonamento dinâmico é a implementação que permite obter melhores desempenhos, principalmente, para imagens de grandes dimensões.

Podemos concluir ainda que para as implementações de Tiling não foram obtidos melhores resultados devido à *bandwidth* da memória.

Apêndice A

Resultados Obtidos

A.1 Sequencial

Image	Simplista	Tiling	Tiling otimizado
Floppy Disk	8e-6	17e-6	17e-6
Finger Print	0.445	0.887	0.792
Gun	29.0	141.6	13.3

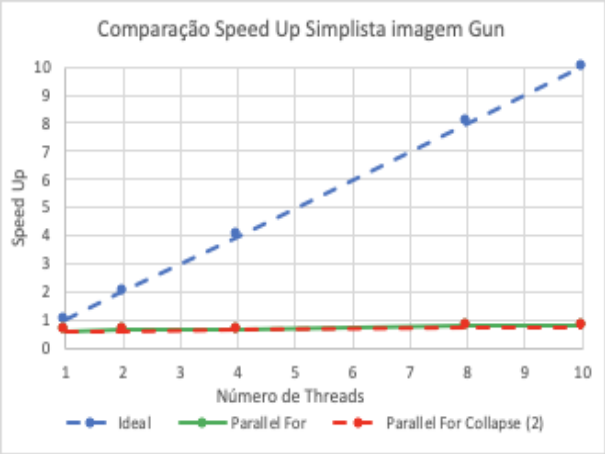
Tabela A.1: Tempos Obtidos em Sequencial em segundos

A.2 Paralelo

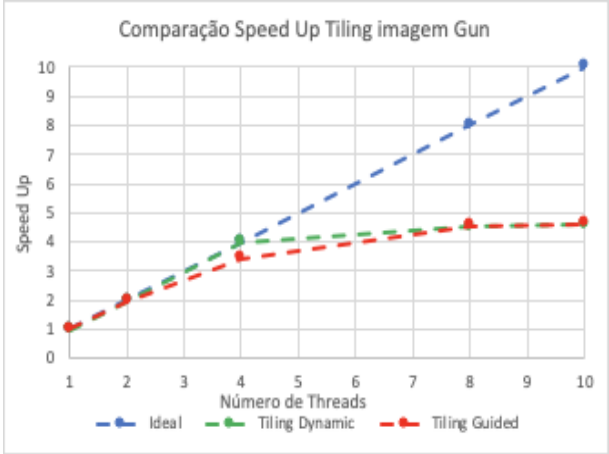
A.2.1 Gun

#T	PF	SUp	TG	SUp	TOG	SUp	TD	SUp	TOD	SUp
1	46.9	0.62	141.8	1.0	13.7	0.97	142.8	0.99	13.7	0.97
2	44.6	0.65	71.9	1.97	7.4	1.80	72.1	1.96	7.1	1.87
4	44.9	0.65	41.5	3.41	9.1	1.46	35.6	3.98	4.3	3.09
8	36.7	0.79	31.1	4.55	5.4	2.46	31.4	4.5	4.3	3.09
10	36.8	0.79	30.7	4.61	4.3	3.09	30.8	4.58	4.3	3.09
16	34.6	0.84	29.4	4.82	4.3	3.09	29.5	4.79	4.3	3.09
20	40.0	0.73	31.5	4.50	4.4	3.02	29.9	4.73	4.5	2.96
32	38.7	0.75	36.1	3.92	5.2	2.56	36.5	3.87	5.5	2.42
40	40.5	0.72	38.3	3.70	5.6	2.38	38.8	3.64	6.0	2.22

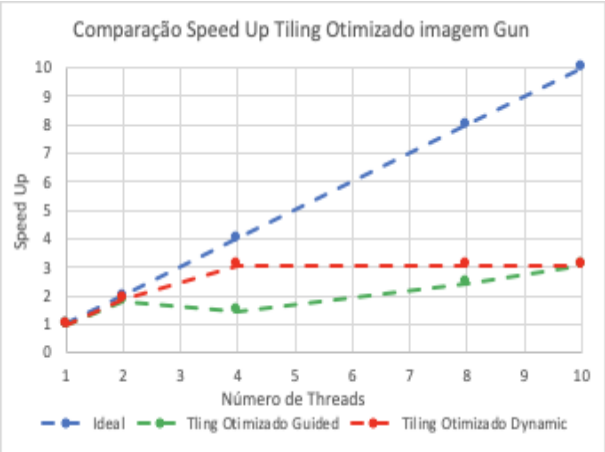
Tabela A.2: Tempos Obtidos em Paralelo para a imagem Gun(8000x4500) em segundos



(a) Curvas de ganho Simplista da imagem Gun



(d) Curvas de ganho Tiling da imagem Gun

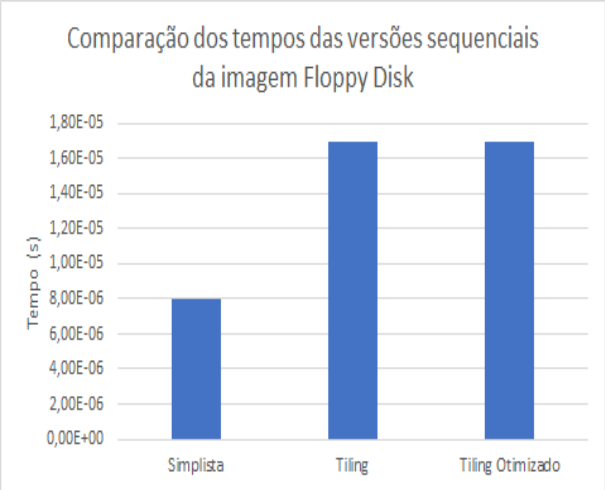


(a) Curvas de ganho Tiling Otimizado da imagem Gun

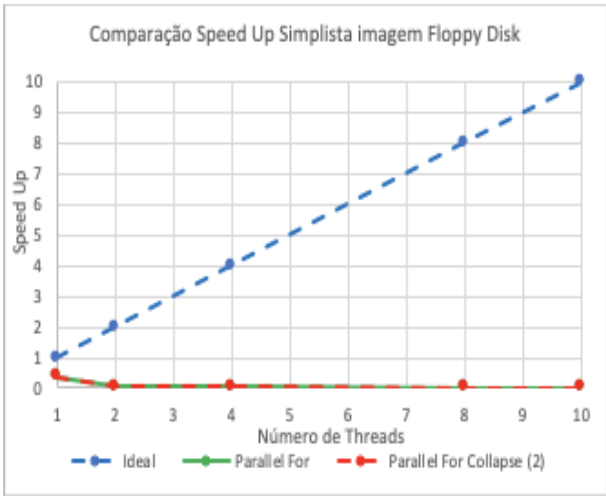
A.2.2 Floppy Disk

#T	PF	SUp	TG	SUp	TOG	SUp	TD	SUp	TOD	SUp
1	0.022	0.36	0.017	1.00	0.017	1.00	0.017	1.00	0.017	1.00
2	0.142	0.06	0.129	0.13	0.098	0.18	0.096	0.18	0.100	0.17
4	0.164	0.05	0.125	0.13	0.135	0.13	0.144	0.12	0.126	0.13
8	0.234	0.03	0.206	0.09	0.220	0.08	0.216	0.08	0.207	0.09
10	0.268	0.03	0.249	0.07	0.251	0.07	0.259	0.07	0.249	0.07
16	0.400	0.02	0.358	0.05	0.354	0.05	0.361	0.05	0.361	0.05
20	0.446	0.02	0.424	0.04	0.450	0.04	0.426	0.04	0.423	0.04
32	0.769	0.01	0.725	0.02	0.726	0.02	0.730	0.02	0.725	0.02
40	15.270	0.00	5.702	0.00	5.750	0.00	5.728	0.00	5.755	0.00

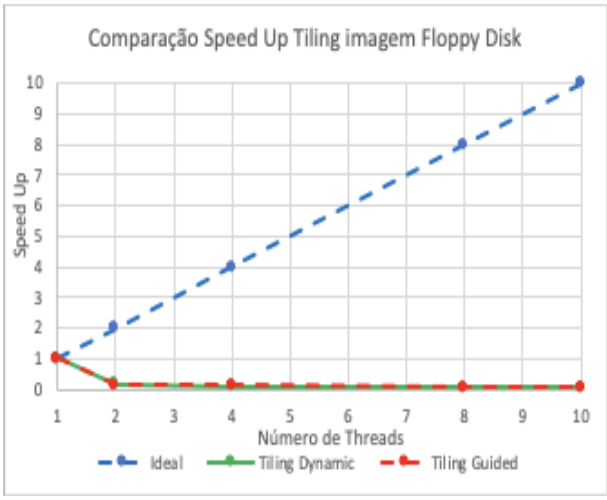
Tabela A.3: Tempos Obtidos em Paralelo para a imagem Floppy Disk(16x16) em milisegundos



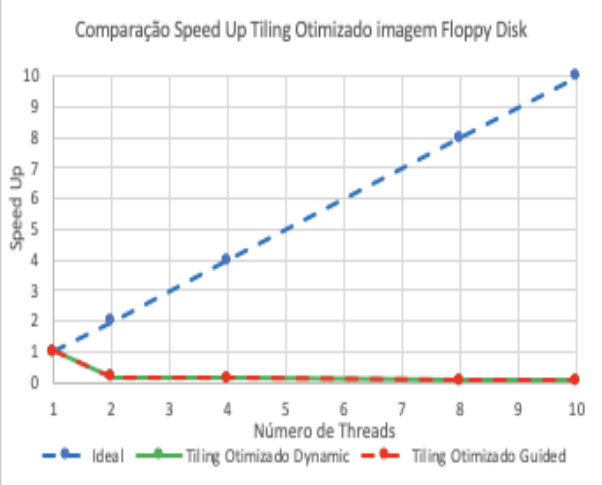
(a) Desempenho das versões sequênciais



(b) Curvas de ganho Simplista da imagem Floppy Disk



(c) Curvas de ganho Tiling da imagem Floppy Disk

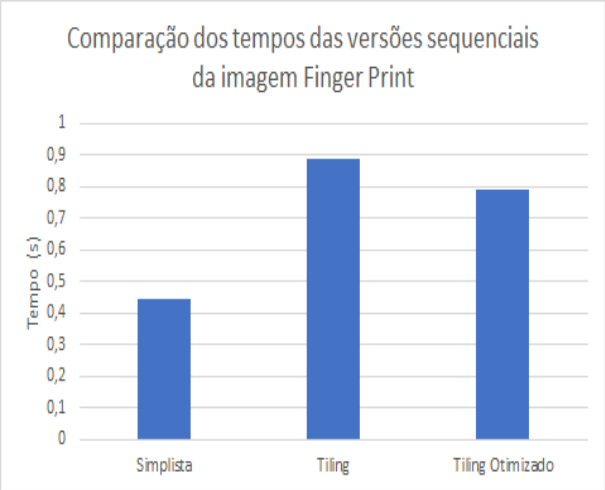


(d) Curvas de ganho Tiling Otimizado da imagem Floppy Disk

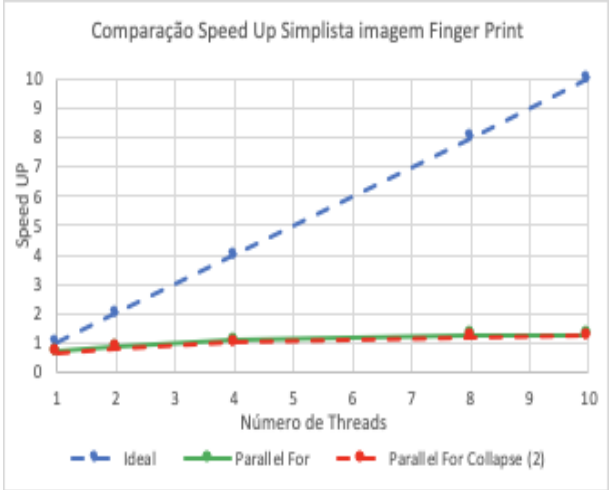
A.2.3 Finger Print

#T	PF	SUp	TG	SUp	TOG	SUp	TD	SUp	TOD	SUp
1	0.618	0.72	0.901	0.98	0.789	1.00	0.898	0.99	0.789	1.00
2	0.511	0.87	0.659	1.35	0.591	1.34	0.653	1.36	0.594	1.33
4	0.417	1.07	0.520	1.71	0.488	1.62	0.528	1.68	0.492	1.61
8	0.350	1.27	0.458	1.94	0.437	1.81	0.460	1.93	0.437	1.81
10	0.350	1.27	0.405	2.19	0.406	1.95	0.408	2.17	0.405	1.96
16	0.318	1.40	0.405	2.19	0.406	1.95	0.408	2.17	0.407	1.95
20	0.316	1.41	0.411	2.16	0.410	1.93	0.411	2.16	0.407	1.95
32	0.319	1.40	0.436	2.03	0.430	1.84	0.429	2.04	0.428	1.85
40	0.468	0.95	0.477	1.86	0.480	1.65	0.480	1.85	0.474	1.67

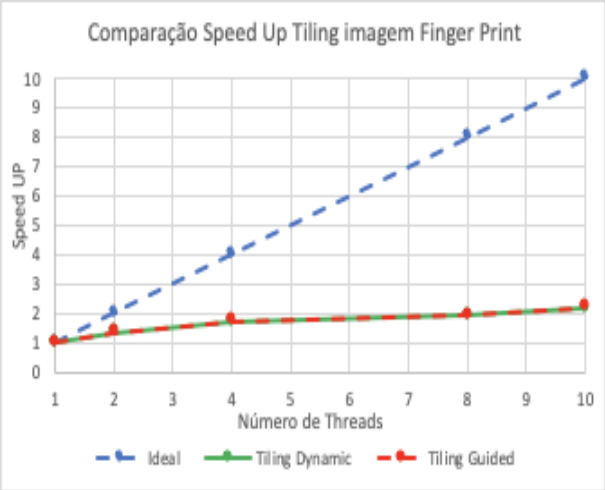
Tabela A.4: Tempos Obtidos em Paralelo para a imagem Finger Print(1592x2312) em segundos



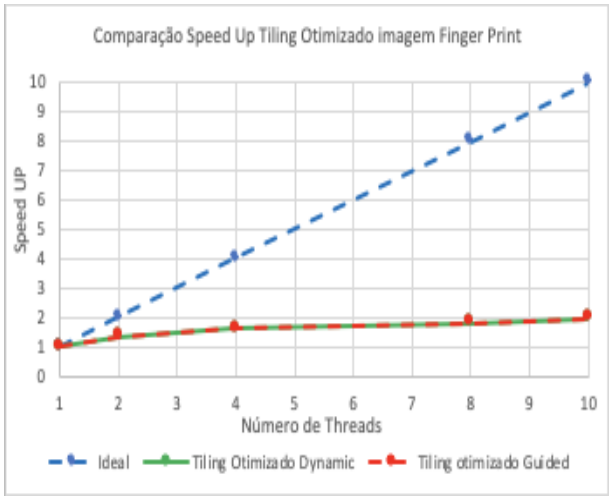
(a) Desempenho das versões sequênciais



(b) Curvas de ganho Simplista da imagem Finger Print

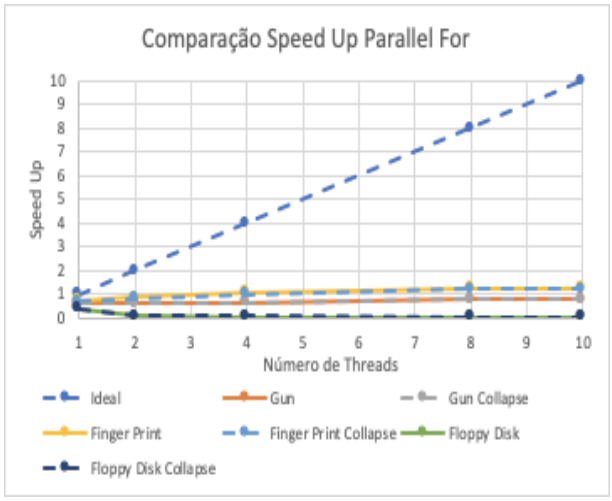


(c) Curvas de ganho Tiling da imagem Finger Print

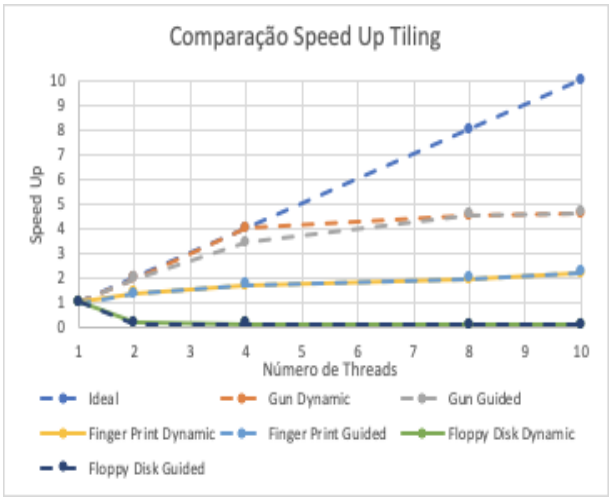


(d) Curvas de ganho Tiling Otimizado da imagem Finger Print

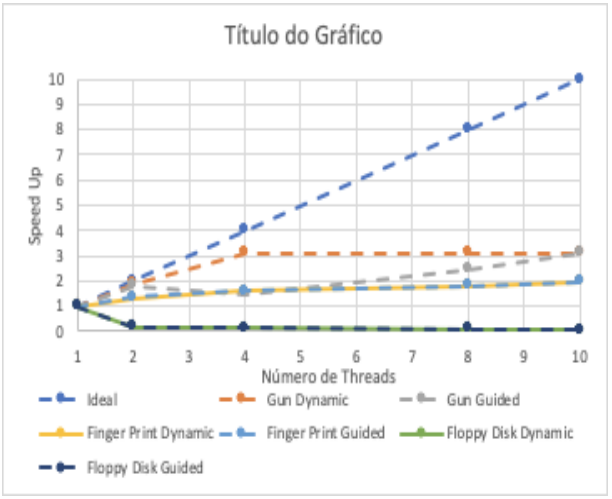
A.2.4 Comparação de Speed Up entre diferentes imagens



(a) Curvas de ganho Parallel For para todas as imagens



(b) Curvas de ganho Tiling para todas as imagens



(c) Curvas de ganho Tiling Otimizado para todas as imagens

A.3 Siglas

Sigla	Significado
#T	Número de Threads Utilizadas
SUp	Speed Up
PF	Parallel For
TG	Tiling Guided
TD	Tiling Dynamic
TOG	Tiling Optimized Guided
TOD	Tiling Optimized Dynamic

Tabela A.5: Significados Siglas das Tabelas Anteriores