MAC 0219/5742 – Introdução à Computação Concorrente, Paralela e Distribuída

Prof. DR. Alfredo Goldman,
Guilherme Souza S.

1

1. Relatório Geral

Para esse trabalho implementou-se de forma simples a reprodução do conjunto de Mandelbrot como pedido no enunciado, podendo ser implementado de varias formas diferentes mas ao mesmo tempo parecidas. Em um primeiro momento realizou-se a implementação sequencia do mesmo, executando-o para diferentes dimensões de imagem iniciando em 500x480 logo passando para dimensão de imagem digital, D-VHS, HDV, 2k, 4k e por fim 8k. Como pode-se observar na Tabela 1.1 ao final do relatório, quando chegamos em uma imagem 8k, o tempo consumido para processamento é um tanto quanto alto se for compararmos com as maneiras paralelizáveis.

Dando seguimento ao trabalho, passou-se para implementação da versão paralelizada com *Opem MP*, onde visivelmente pelo calculo ocorrido se tem maneiras muito eficientes desta paralelização correr, onde ocorreu uma melhora em tempo de execução significante quando comparado a primeira implementação.

Como terceiro e último passo, implementou-se o mesmo trabalho em *Cuda*, onde comprar com a sequencia ficou injusto com tamanha melhora em tempo de execução conforme se aumenta o número de threads por bloco, ainda mais quando pegamos como parâmetro as imagens de dimensão 8k dos 3 testes, *Cuda* mostrou-se superior.

Todos os resultados podem ser visto ao fim do relatório, como tabelas de tempo de execução com os devidos parâmetros, ou seja, entradas dada para calculo do conjunto, dimensão da imagens e número de threads.

2. Relato Pessoal

No início do trabalho parecia que a parte mais complicada seria a parte de cuda, doce engano, particularmente tive dificuldade em desenvolver o trabalho, tendo um grande ajuda dos instrutores, os quais me explicarão inúmeras vezes até que eu conseguisse entender e programar. O trabalho em si era muito interessante, pela forma que ele funciona e o conhecimento adquirido em imagens fractais o sobre o conjunto em si abrindo portas para conhecimentos de outros conjuntos como o de Júlia e outros.

Primeiramente entendi o caculo com um bom exemplo dado na internet, onde consegui entender em si, toda essa parte foi desenvolvida no papel e não computacionalmente, fazendo uso de uma matriz 8x8. Logo depois passei a procurar alguns exemplos na internet de modo que conseguisse entender a forma correta para programar, porém implementando somente ela em ASCII, onde era possível se ter uma visão geral de como ficaria a imagem pelo terminal, porém tendo somente duas opções, ou seja diverge ou não (”*” ou ”0”) no caso da implementação que optei para entendimento do mesmo.

Logo após essa parte, passei para parte de usar a LIBPNG no caso para uso da png.h para formar a imagem. Olhando alguns Tutoriais observei que havia pouca diferença de uso entre para ela, sendo disponibilizado na própria pagina da lib um exemplo de como usá-la e o exemplo era justamente uma imagem fractal, onde foi de muita ajuda na parte de implementação da png, logo fiz o reaproveitamento da mesma para coloração da minha imagem, porém após alguns problemas com o calculo realizado para geração das cores, partindo da dica do monitor, deixei cores fixas para numero de iterações executadas, o que gerou um imagem razoavelmente bonita, nada comparada com a do exemplo, porém bela perante a dificuldade que tive.

Com a implementação em OPEN_MP inicialmente tive problemas em gerar a imagem corretamente, pelo fato da forma que estava tentando implementar o paralelismo, fazendo uso somente do *parallel for*, porém após observar um pouco entendi que muitas das variáveis as quais então executando não poderiam ser simplesmente acessada a qualquer momento, até pela dependência gerada pelas mesmas logo optei por privatizar elas para uso e, organizar os *for* para uso do *collapse*.

Dando continuidade, quando foi para a parte de programar em *Cuda*, demorei para entender a questão de blocos mesmo tendo visto isso no trabalho anterior, porém agora ”linkando” com a imagem parecia complicado de entender no meu caso. Novamente com a ajuda dos instrutores consegui ver isso acontecendo de fato, logo para um solução simples a qual encontrei, resolvi jogar o calculo para ser feito na placa de vídeo, onde de fato a paralelização fazia sentido para mim, assim que acabava bastava puxar o buffer para o host e realizar a ”impressão da imagem”, sendo ela finalizado no host como dito, um forma simples de se fazer o uso da placa de vídeo, havendo implementações melhores pelo que pude observar na internet, porém pelo tempo e pela minha dificuldade aceitei como uma boa solução.

3. Hardware

O computador no qual os testes foram rodados conta com as configurações apresentadas abaixo, sendo dividida em duas máquinas de execução, pelo fato da necessidade da placa de vídeo NVIDIA exigida pelo trabalho, tal diferença entre hardware foi desconsiderada nos testes. Tais informações foram obtidas com o comando *lscpu* em conjunto com *lshw* e *nvidia-smi*.

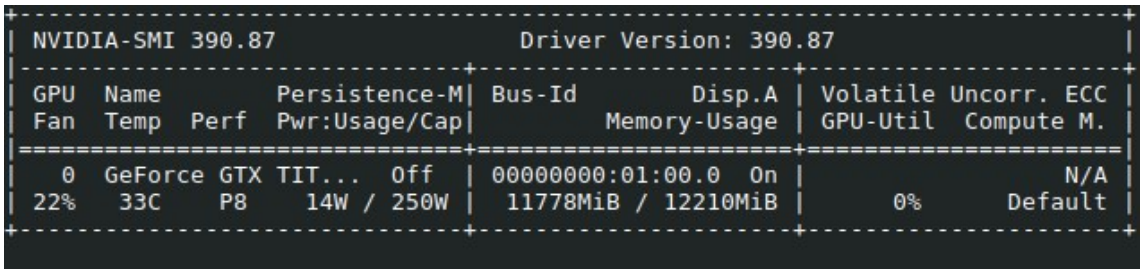

Figure 1. Informações da Placa de Vídeo

```
Arquitetura:                    x86_64
Modo(s) operacional da CPU:     32-bit, 64-bit
Ordem dos bytes:                Little Endian
CPU(s):                         4
Lista de CPU(s) on-line:        0-3
Thread(s) per n cleo:           2
N cleo(s) por soquete:          2
Soquete(s):                     1
N (s) de NUMA:                  1
ID de fornecedor:               GenuineIntel
Fam lia da CPU:                 6
Modelo:                         58
Nome do modelo:                 Intel(R) Core(TM) i3-3217U CPU @ 1.80GHz
Step:                           9
CPU MHz:                        1356.701
CPU MHz m x.:                   1800,0000
CPU MHz m n.:                   800,0000
BogoMIPS:                       3591.71
Virtualiza o:                   VT-x
cache de L1d:                   32K
cache de L1i:                   32K
cache de L2:                    256K
cache de L3:                    3072K
CPU(s) de n 0 NUMA:             0-3
```

Listing 1. informações CPU (Sem placa de Video)

```
*-cache:0
             description: L1 cache
             physical id: 6
             slot: L1-Cache
             size: 32KiB
             capacity: 32KiB
             capabilities: internal write-through instruction
             configuration: level=1
        *-cache:1
             description: L2 cache
             physical id: 7
             slot: L2-Cache
             size: 256KiB
             capacity: 256KiB
             capabilities: internal write-through unified
             configuration: level=2
        *-cache:2
             description: L3 cache
             physical id: 8
             slot: L3-Cache
             size: 3MiB
             capacity: 3MiB
             capabilities: internal write-back unified
             configuration: level=3
```

Listing 2. Informacoes Cache (Sem placa de Video)

```
*-memory
          description: System Memory
          physical id: 29
          slot: System board or motherboard
          size: 4GiB
```

Listing 3. Informacoes Memoria (Sem placa de video)

```
Arquitetura:             x86_64
Modo(s) operacional da CPU:32-bit, 64-bit
Ordem dos bytes:         Little Endian
CPU(s):                  8
Lista de CPU(s) on-line:0-7
Thread(s) per n cleo: 2
N cleo(s) por soquete:4
Soquete(s):              1
N (s) de NUMA:           1
ID de fornecedor:        GenuineIntel
Fam lia da CPU:          6
Modelo:                  58
Nome do modelo:          Intel(R) Xeon(R) CPU E3-1230 V2 @ 3.30GHz
Step:                    9
CPU MHz:                 1985.559
CPU MHz m x.:            3700,0000
CPU MHz m n.:            1600,0000
BogoMIPS:                6585.24
Virtualiza o:            VT-x
cache de L1d:            32K
cache de L1i:            32K
cache de L2:             256K
cache de L3:             8192K
CPU(s) de n 0 NUMA:      0-7
```

Listing 4. informações CPU (Com placa de Video

4. Resultados

Em um primeiro momento executou-se 15 vezes cada algoritmo com dimensões variadas seguindo um padrão de imagem sendo a ultima execução com resolução 8k, as threads estão em ordem múltipla de 2 indo de 2 a 256 para que tais diferenças possam realmente observadas. Realizando a variação de numero de threads em mesma quantidade em cada algoritmo paralelizado, de forma que fosse possível avaliar o desempenho em numero de threads em questão de tamanho e forma de paralelização, usou-se o tempo real através do comando *time* do GNU, realizando algus ajustes na variável de forma a devolver somente o valor desejado. Para manipulação de tais dados usou-se o meio estatístico do intervalo de segurança de forma a ser possível criar gráficos com uma visibilidade e resultados mais demonstrativos, fazendo a escolha de 95% de confiança, seguindo o modelo padrão de cálculo do intervalo, como pode ser observado nos resultados ao final do relatório.

Quando observado os tempos de paralelização com *open MP* pode-se observar uma boa melhora no tempo, ainda mais quando se há o aumento das dimensões da imagem, porem pelos testes executados observes que a partir de 16 threads o tempo de fato não tem uma melhora considerável, creio que isso ocorra pelo método de implementação

realizado por mim, pelo fato de privatizar uma grande parte de variáveis, as threads tem que esperar um tempo para acessa lá e conforme o numero de threads aumenta o overhead da espera acaba não diluindo nas dimensões utilizadas no testes.

Table 1. Comparação entre os algoritmos em tempo de execução para imagem de dimensão 3840x2160

Algoritimo	Numero de Threads	Tempo de Execução
Sequêncial	-	5,896
Opem_MP	2	3,79
Cuda_NVIDIA	2	0,774

Quando passamos a observar os resultados obtidos com a paralelização na placa de vídeo, pode-se observar que os tempos baixaram bastante em consideração a implementação de *open MP* como pode ser visto na Tabela 0 onde comparamos o tempo de execução da imagem com dimensão de 3840x2160, e em comparação entre as execuções consigo mesmo e o aumento das dimensões e numero de threads por bloco, é possível notar que com 128 threads obteve-se o melhor resultado em muitos casos, em outros se equiparando com o de 64 e 256, independente do tamanhos (dentro dos selecionados para execução dos testes), logo podemos ver que a execução na placa de vídeo obteve um ganho de tempo realmente favorável.

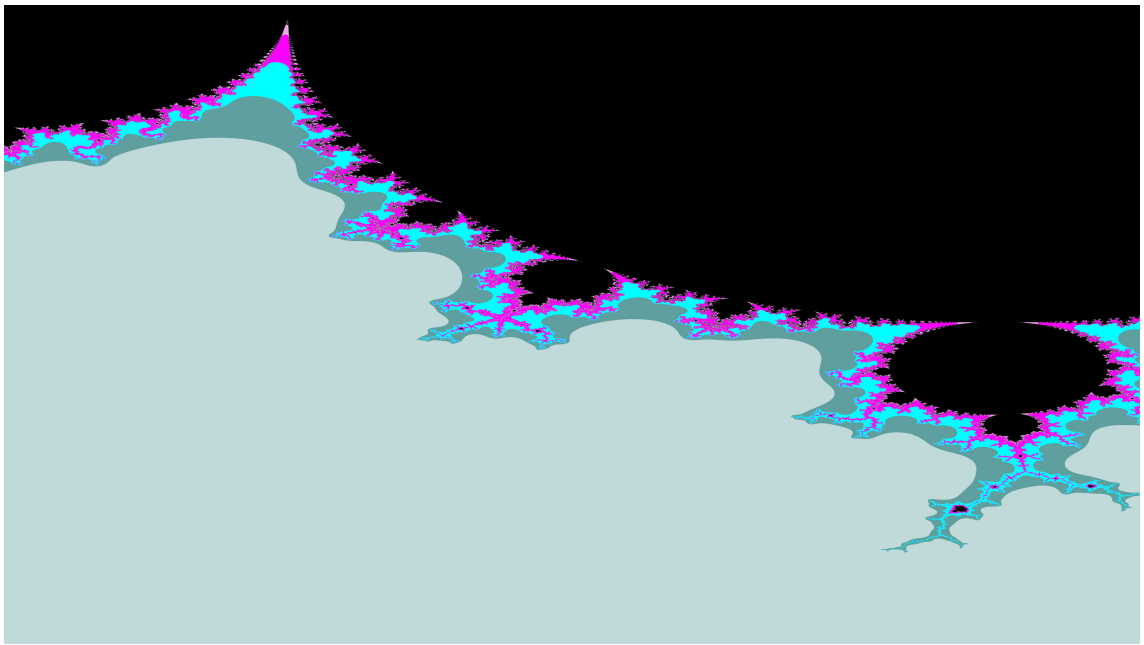

Figure 2. Imagem gerada para testes, com entradas (0.0, -1,0)(0.0, -1.32)

Tabela 1.1: Tempo de execução do codigo sequêncial							
Tamanho da Imagem	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
500x480	0,17	0,006197541755	0,18	0,17	0,003136331235	0,172	0,18
720x576	0,292	0,002390457219	0,2943904572	0,2896095428	0,001209716035	0,2905	0,293
1280x720	0,686	0,03309898574	0,7190989857	0,6529010143	0,01675009011	0,647	0,708
1440x1080	1,182	0,04959473858	1,231594739	1,132405261	0,02509793945	1,13	1,209
2080x1080	1,543	0,02789128555	1,570891286	1,515108714	0,01411467861	1,537	1,563
3840x2160	5,896	0,2641643067	6,160164307	5,631835693	0,1336831278	5,7435	6,133
7680x4320	23,13	0,2203717638	23,35037176	22,90962824	0,1115214505	22,974	23,3345

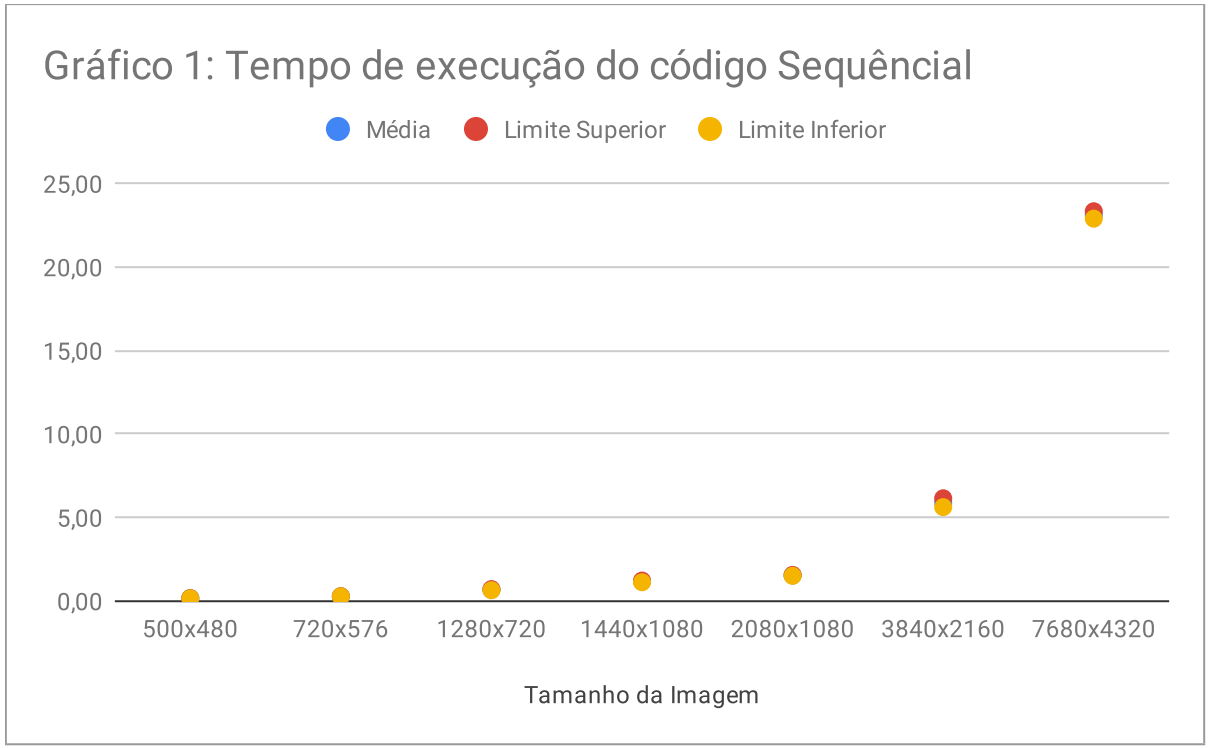

Tabela 2: Tempo de execução do codigo Paralelizado com CUDA_NVIDIA com resolução de 500x480							
Numero de Threads	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	0,118	0,004272837799	0,1222728378	0,1137271622	0,002162314541	0,117	0,121
4	0,113	0,002789435854	0,1157894359	0,1102105641	0,001411623373	0,112	0,115
8	0,112	0,003376388603	0,1153763886	0,1086236114	0,00170865699	0,108	0,1135
16	0,11	0,002658320272	0,1126583203	0,1073416797	0,001345270953	0,11	0,112
32	0,11	0,002463060427	0,1124630604	0,1075369396	0,001246457653	0,109	0,1135
64	0,109	0,003363671463	0,1123636715	0,1056363285	0,001702221347	0,1075	0,1125
128	0,108	0,002971291206	0,1109712912	0,1050287088	0,001503653188	0,107	0,1115
256	0,112	0,009033271833	0,1210332718	0,1029667282	0,004571382287	0,109	0,114

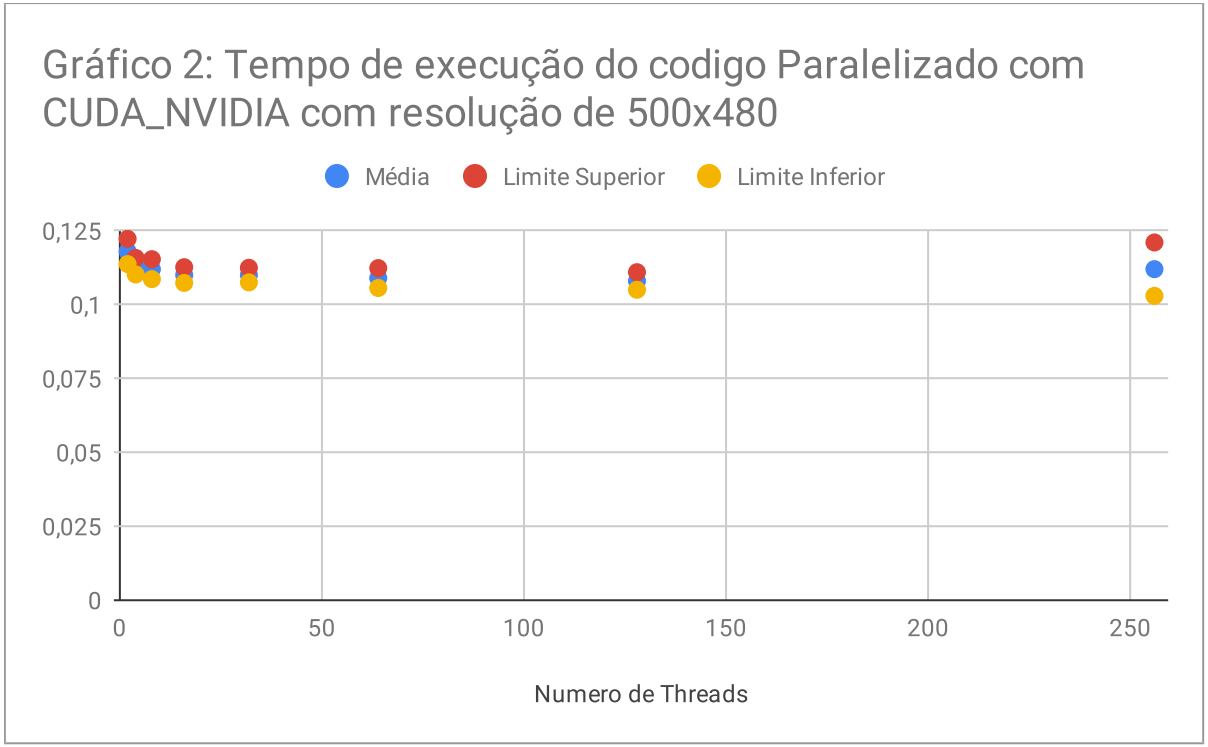

Tabela 3: Tempo de execução do codigo Paralelizado com CUDA_NVIDIA com resolução de 720x576							
Numero de Threads	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	0,138	0,004257206775	0,1422572068	0,1337427932	0,002154404296	0,135	0,14
4	0,126	0,002478478796	0,1284784788	0,1235215212	0,001254260281	0,125	0,129
8	0,123	0,003754362542	0,1267543625	0,1192456375	0,00189993468	0,1205	0,1255
16	0,119	0,00402610529	0,1230261053	0,1149738947	0,002037452958	0,118	0,1245
32	0,12	0,004610546915	0,1246105469	0,1153894531	0,002333215793	0,118	0,1245
64	0,118	0,003811011614	0,1218110116	0,1141889884	0,001928602539	0,1175	0,121
128	0,119	0,002374467359	0,1213744674	0,1166255326	0,001201624199	0,118	0,122
256	0,121	0,003788453636	0,1247884536	0,1172115464	0,001917186837	0,1175	0,122

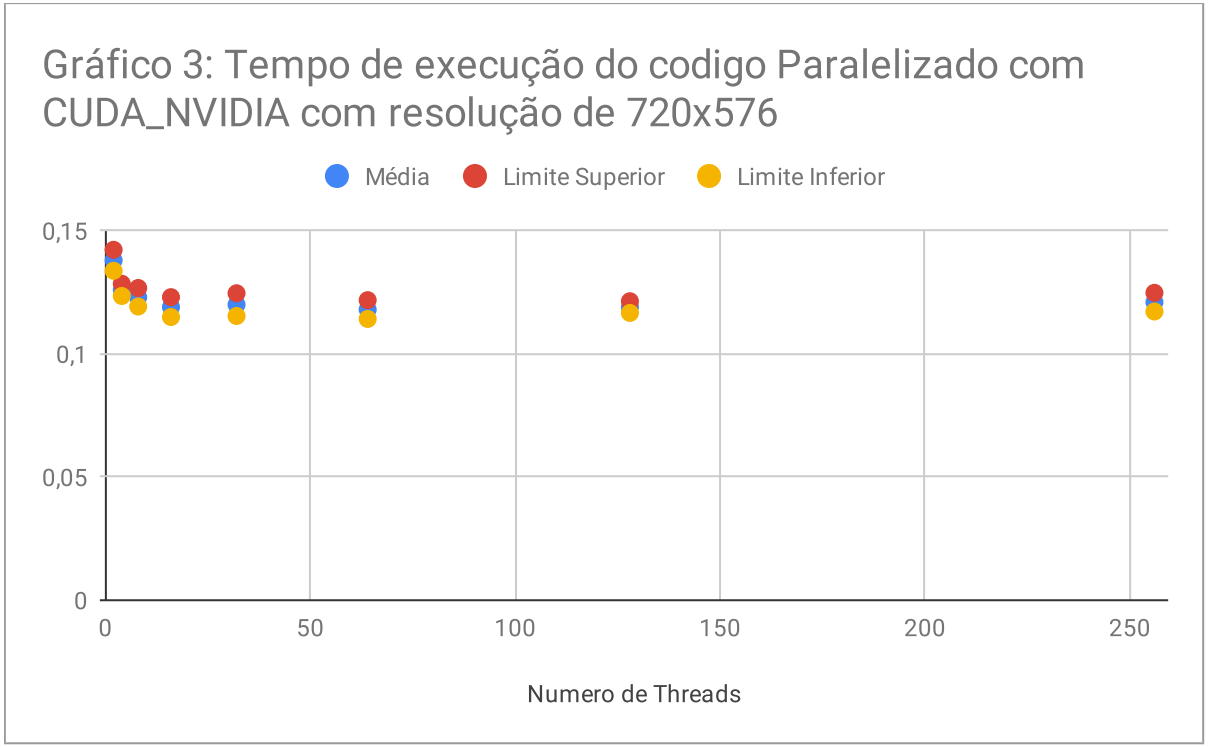

Tabela 4: Tempo de execução do codigo Paralelizado com CUDA_NVIDIA com resolução de 1280x720							
Numero de Threads	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	0,177	0,003312314685	0,1803123147	0,1736876853	0,001676231709	0,176	0,181
4	0,16	0,004431489054	0,1644314891	0,1555685109	0,002242601678	0,1585	0,163
8	0,152	0,00511580185	0,1571158019	0,1468841981	0,002588905371	0,1505	0,1575
16	0,146	0,002725540575	0,1487255406	0,1432744594	0,001379288495	0,144	0,1475
32	0,143	0,003058166273	0,1460581663	0,1399418337	0,00154761723	0,143	0,146
64	0,143	0,002531703737	0,1455317037	0,1404682963	0,001281195322	0,143	0,146
128	0,144	0,004642454195	0,1486424542	0,1393575458	0,002349362808	0,142	0,1485
256	0,144	0,006419464449	0,1504194644	0,1375805356	0,00324863755	0,143	0,1485

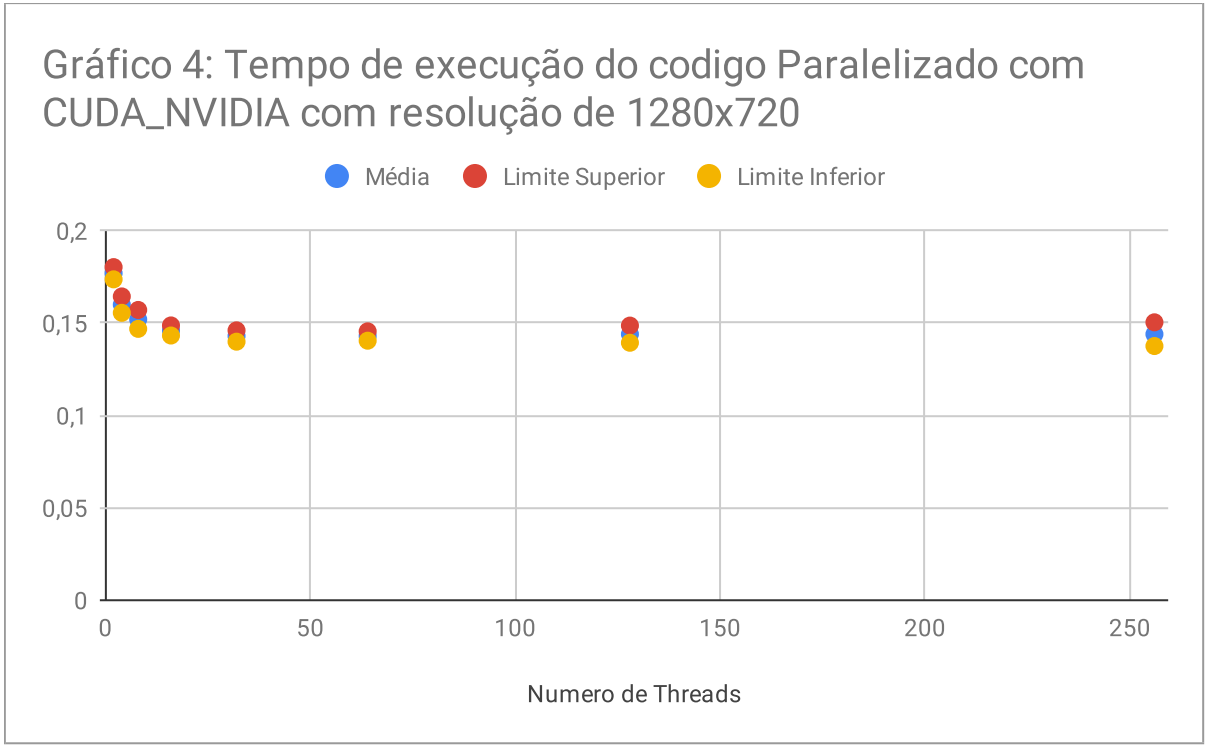

Tabela 5: Tempo de execução do codigo Paralelizado com CUDA_NVIDIA com resolução de 1440x1080							
Numero de Threads	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	0,226	0,004778921974	0,230778922	0,221221078	0,002418423764	0,225	0,2285
4	0,2	0,005539468773	0,2055394688	0,1944605312	0,002803306476	0,1985	0,2025
8	0,185	0,003673586544	0,1886735865	0,1813264135	0,001859057136	0,183	0,1875
16	0,18	0,004549201918	0,1845492019	0,1754507981	0,002302171512	0,176	0,1835
32	0,174	0,003885259077	0,1778852591	0,1701147409	0,001966176249	0,173	0,177
64	0,175	0,004186144948	0,1791861449	0,1708138551	0,002118442711	0,174	0,178
128	0,175	0,0075561391	0,1825561391	0,1674438609	0,003823863721	0,173	0,178
256	0,175	0,004956957592	0,1799569576	0,1700430424	0,002508520562	0,173	0,1815

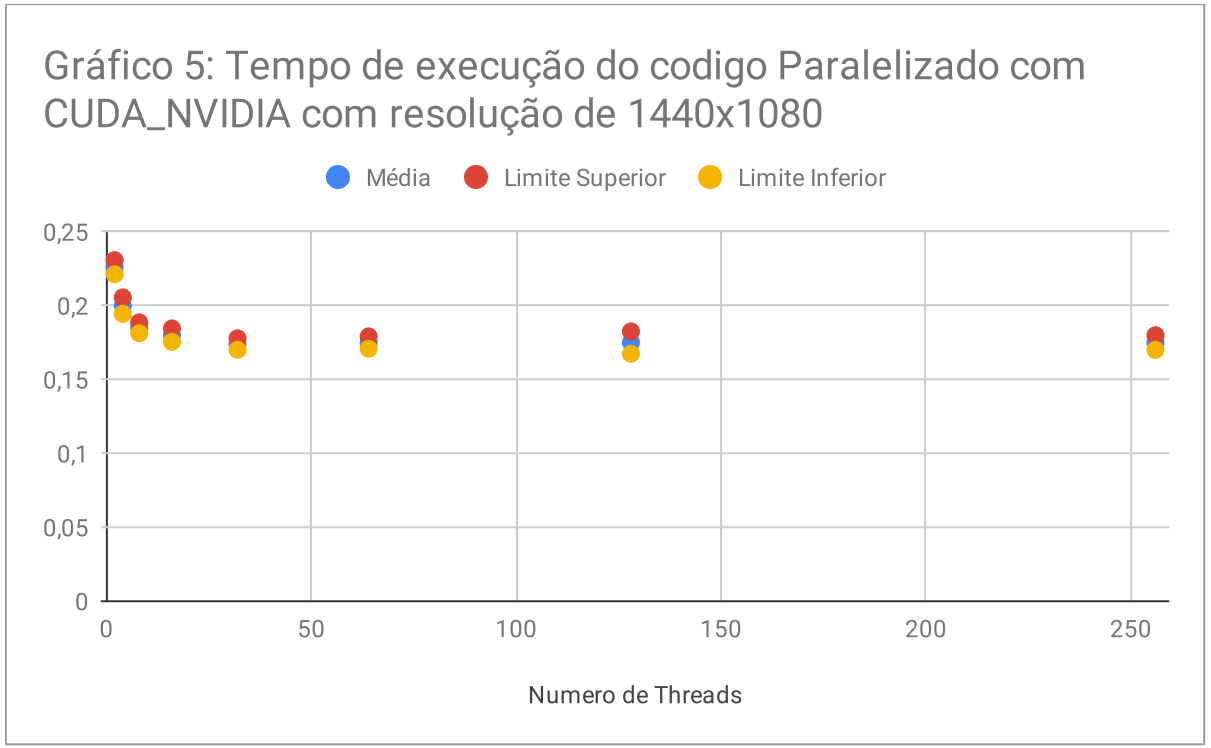

Tabela 6: Tempo de execução do codigo Paralelizado com CUDA_NVIDIA com resolução de 2048x1080							
Numero de Threads	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	0,279	0,004988558337	0,2839885583	0,2740114417	0,002524512452	0,278	0,283
4	0,24	0,002774029217	0,2427740292	0,2372259708	0,001403826682	0,2395	0,241
8	0,22	0,002737743249	0,2227377432	0,2172622568	0,001385463786	0,218	0,221
16	0,213	0,007971974721	0,2209719747	0,2050280253	0,004034301714	0,211	0,2245
32	0,207	0,005590808784	0,2125908088	0,2014091912	0,002829287629	0,205	0,2145
64	0,208	0,004511361319	0,2125113613	0,2034886387	0,002283021878	0,205	0,212
128	0,208	0,003997618339	0,2119976183	0,2040023817	0,002023036836	0,2065	0,21
256	0,209	0,01167333143	0,2206733314	0,1973266686	0,005907412234	0,207	0,2135

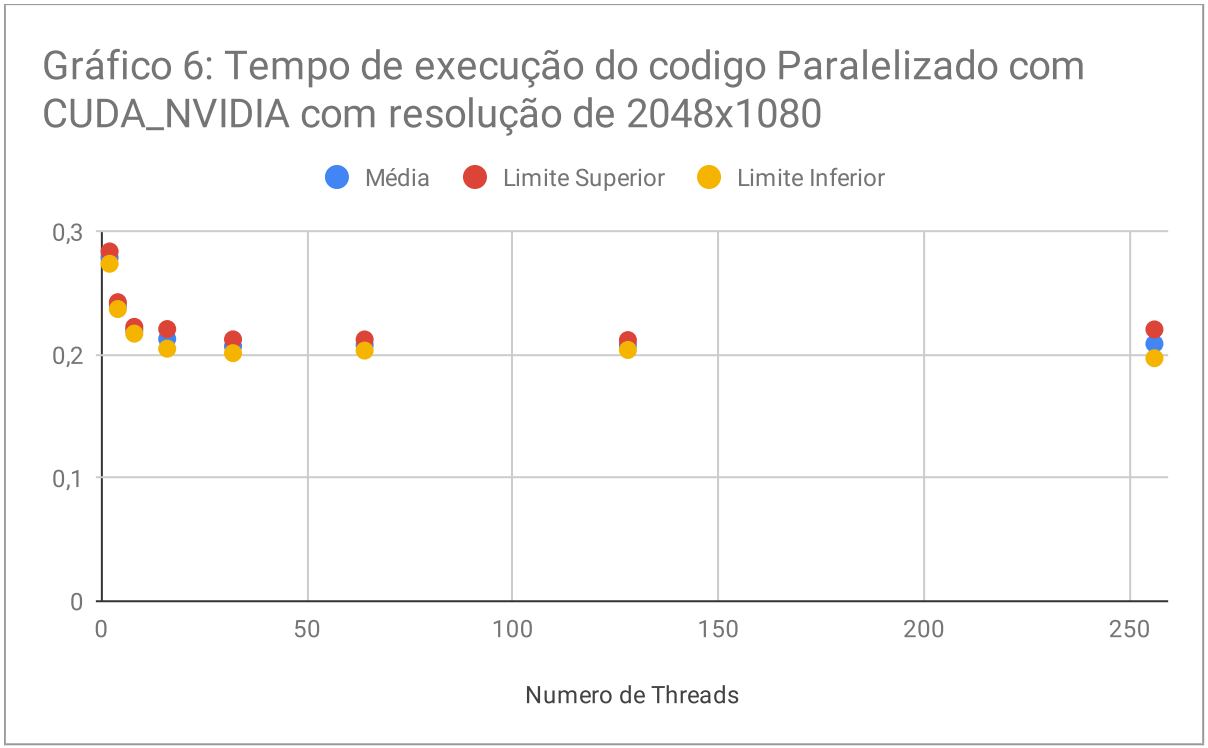

Tabela 7: Tempo de execução do codigo Paralelizado com CUDA_NVIDIA com resolução de 3840x2160							
Numero de Threds	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	0,774	0,01356465997	0,78756466	0,76043534	0,006864538946	0,7705	0,778
4	0,629	0,00773489311	0,6367348931	0,6212651069	0,003914324069	0,6245	0,6315
8	0,56	0,007539104406	0,5675391044	0,5524608956	0,003815243135	0,5565	0,5615
16	0,52	0,01101081719	0,5310108172	0,5089891828	0,005572139931	0,5185	0,5245
32	0,507	0,01161772046	0,5186177205	0,4953822795	0,005879269715	0,4995	0,51
64	0,508	0,00959315232	0,5175931523	0,4984068477	0,004854715701	0,5035	0,51
128	0,507	0,01119948978	0,5181994898	0,4958005102	0,005667619682	0,505	0,512
256	0,506	0,01240199677	0,5184019968	0,4935980032	0,006276160997	0,503	0,5145

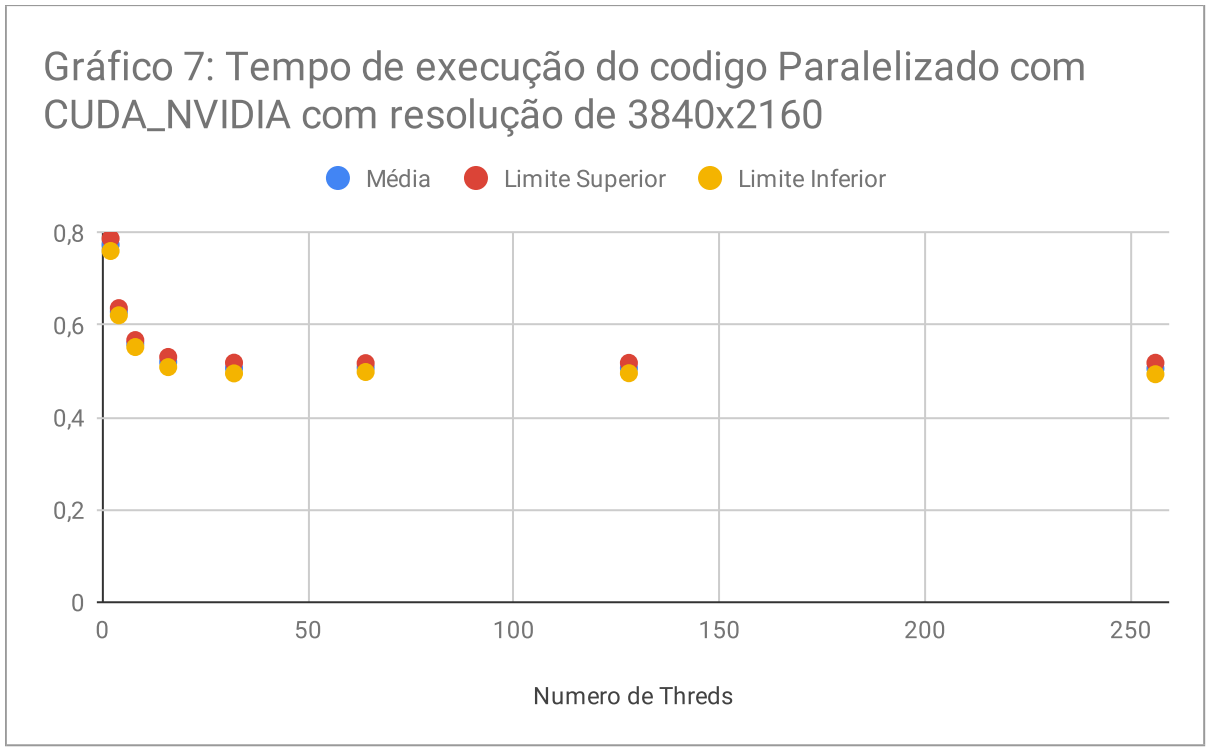

Tabela 8: Tempo de execução do codigo Paralelizado com CUDA_NVIDIA com resolução de 7680x4320							
Numero de Threads	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	2,781	0,04131839208	2,822318392	2,739681608	0,02090960719	2,7725	2,798
4	2,205	0,0132064198	2,21820642	2,19179358	0,006683247737	2,1985	2,2135
8	1,913	0,007435436388	1,920435436	1,905564564	0,003762780844	1,909	1,9175
16	1,768	0,01189517709	1,779895177	1,756104823	0,006019679567	1,7555	1,7745
32	1,698	0,01374495578	1,711744956	1,684255044	0,006955779543	1,687	1,705
64	1,694	0,02387906036	1,71787906	1,67012094	0,01208424983	1,6875	1,7
128	1,69	0,01681326007	1,70681326	1,67318674	0,008508527222	1,6815	1,702
256	1,698	0,01759328876	1,715593289	1,680406711	0,008903268951	1,6885	1,705

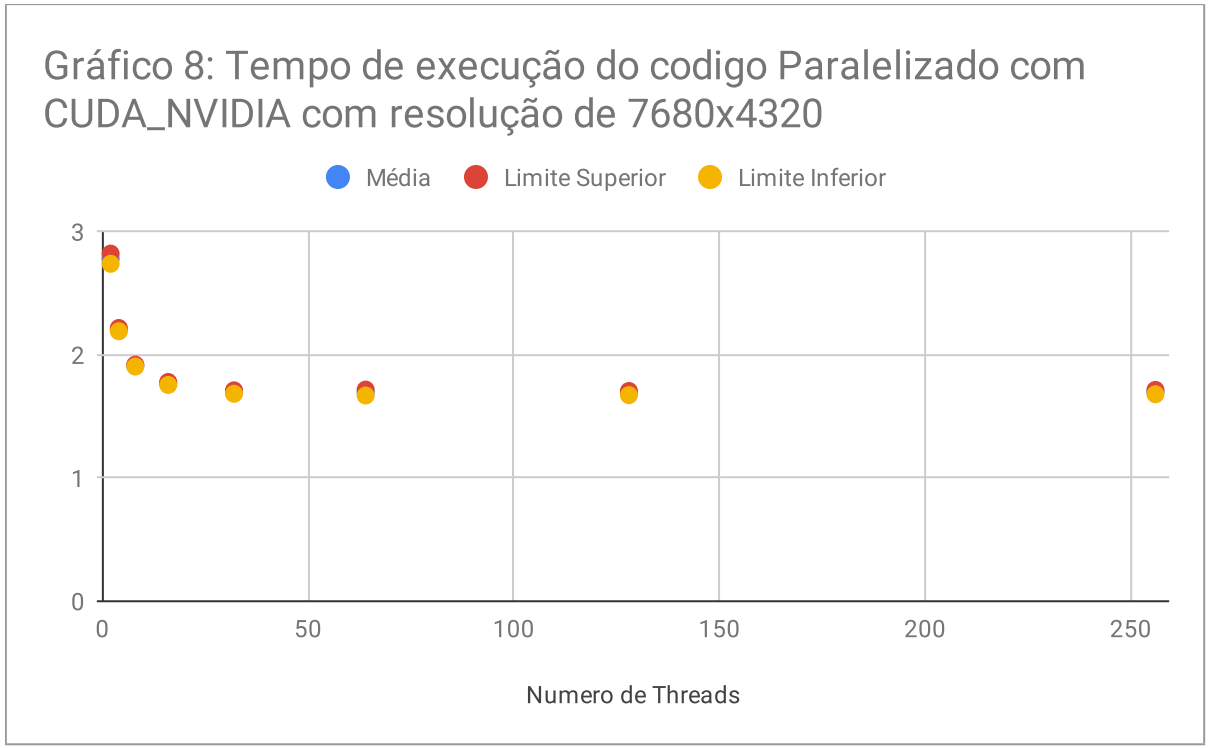

Tabela 10: Tempo de execução do codigo Paralelizado com OPEN_MP resolução de 500x480							
Threads	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	0,11	0,008164965809	0,1181649658	0,1018350342	0,004131966885	0,11	0,12
4	0,1	0,007988086367	0,1079880864	0,09201191363	0,004042455182	0,09	0,105
8	0,1	0,01099783528	0,1109978353	0,08900216472	0,0055655703	0,095	0,11
16	0,09	0,03104528183	0,1210452818	0,05895471817	0,01571079163	0,09	0,095
32	0,09	0,005070925528	0,09507092553	0,08492907447	0,002566195236	0,09	0,1
64	0,1	0,006172133998	0,106172134	0,093827866	0,003123473373	0,09	0,1
128	0,1	0,009411239481	0,1094112395	0,09058876052	0,004762656794	0,095	0,11
256	0,1	0,004879500365	0,1048795004	0,09512049964	0,002469322517	0,1	0,11

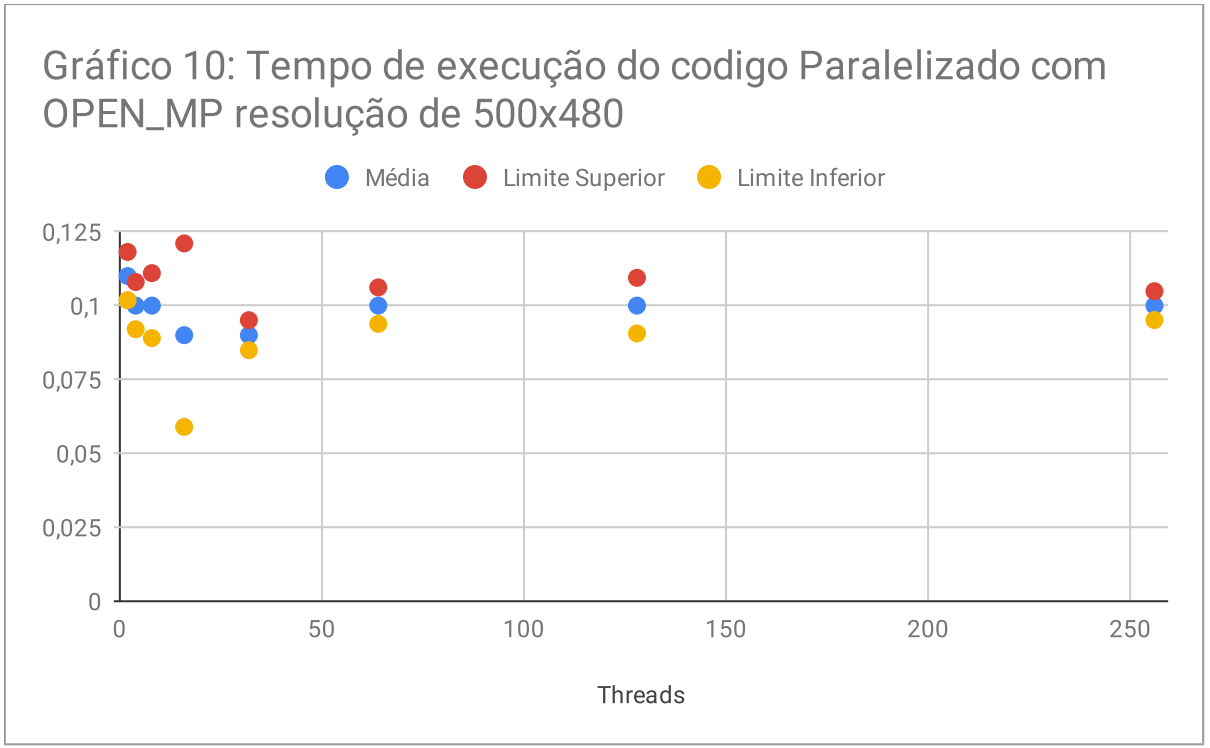

Tabela 11: Tempo de execução do codigo Paralelizado com OPEN_MP com resolução de 720x576							
Tamanho da Imagem	Média	Desvio Padrão	Limite Superior	Limite Inferior	Intervalo de Confiança	Primeiro Quartil	Terceiro Quatil
2	0,19	0,005936168397	0,1959361684	0,1840638316	0,003004060496	0,19	0,2
4	0,16	0,006172133998	0,166172134	0,153827866	0,003123473373	0,16	0,165
8	0,16	0,004140393356	0,1641403934	0,1558596066	0,002095289636	0,16	0,16
16	0,16	0,01099783528	0,1709978353	0,1490021647	0,0055655703	0,16	0,165
32	0,17	0,01302013093	0,1830201309	0,1569798691	0,006588974298	0,16	0,19
64	0,18	0,01447493729	0,1944749373	0,1655250627	0,007325194366	0,165	0,18
128	0,17	0,01032795559	0,1803279556	0,1596720444	0,00522657063	0,17	0,18
256	0,18	0,01780850519	0,1978085052	0,1621914948	0,009012181489	0,175	0,195

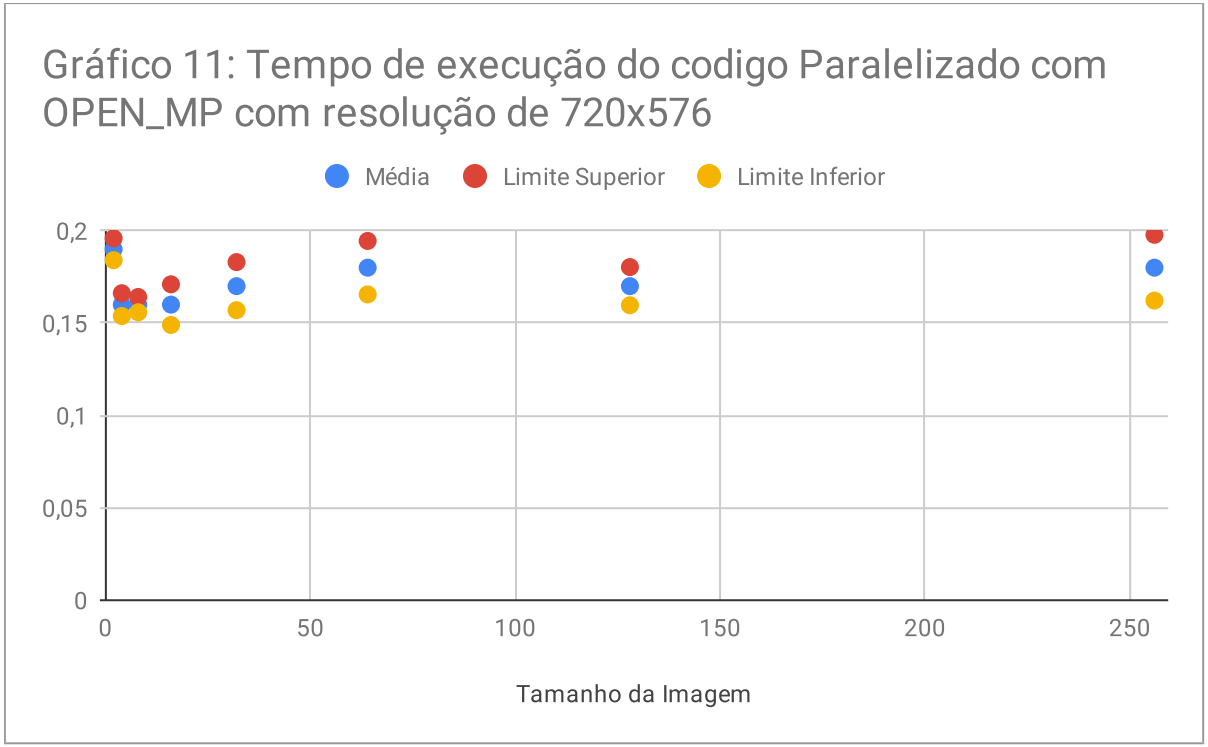

Tabela 12: Tempo de execução do codigo Paralelizado com OPEN_MP com resolução de 1280x720							
Tamanho da Imagem	Média	Desvio Padrão	Limite Superior	Limite Inferior	Intervalo de Confiança	Primeiro Quartil	Terceiro Quatil
2	0,42	0,006399404734	0,4263994047	0,4136005953	0,003238486121	0,42	0,43
4	0,38	0,02250925735	0,4025092574	0,3574907426	0,0113910466	0,35	0,385
8	0,35	0,01187233679	0,3618723368	0,3381276632	0,006008120993	0,35	0,36
16	0,35	0,012344268	0,362344268	0,337655732	0,006246946745	0,35	0,36
32	0,36	0,01764733893	0,3776473389	0,3423526611	0,008930621608	0,355	0,37
64	0,36	0,03326659987	0,3932665999	0,3267334001	0,016834913	0,35	0,375
128	0,36	0,026367368	0,386367368	0,333632632	0,0133434841	0,36	0,38
256	0,36	0,01624221425	0,3762422143	0,3437577857	0,008219543474	0,35	0,37

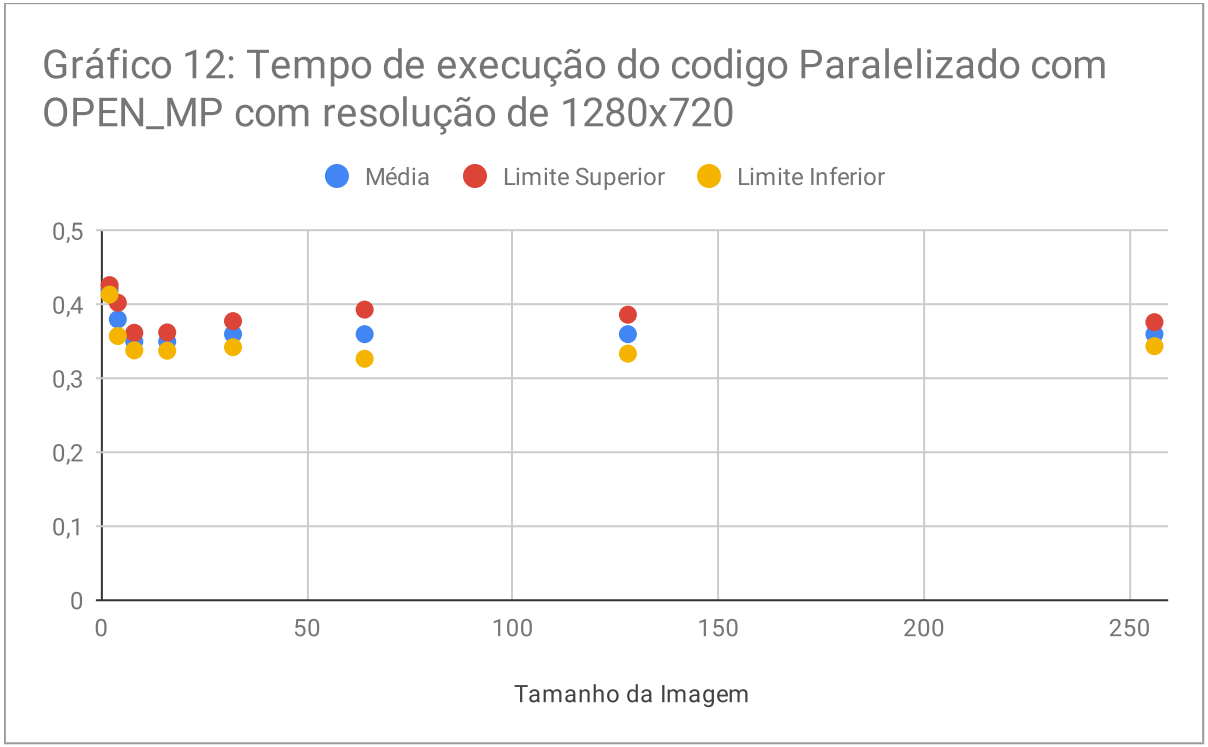

Tabela 13: Tempo de execução do codigo Paralelizado com OPEN_MP com resolução de 1440x1080							
Tamanho da Imagem	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	0,71	0,008837151017	0,718837151	0,701162849	0,004472133285	0,7	0,71
4	0,59	0,02416215065	0,6141621506	0,5658378494	0,01222751064	0,57	0,605
8	0,58	0,01505545305	0,5950554531	0,5649445469	0,007618970479	0,58	0,59
16	0,59	0,02016597795	0,6101659779	0,5698340221	0,01020520539	0,58	0,61
32	0,61	0,02350278606	0,6335027861	0,5864972139	0,01189383225	0,6	0,64
64	0,59	0,01533747356	0,6053374736	0,5746625264	0,007761689925	0,58	0,6
128	0,59	0,01502379066	0,6050237907	0,5749762093	0,00760294739	0,59	0,605
256	0,6	0,02138089935	0,6213808994	0,5786191006	0,01082002915	0,595	0,625

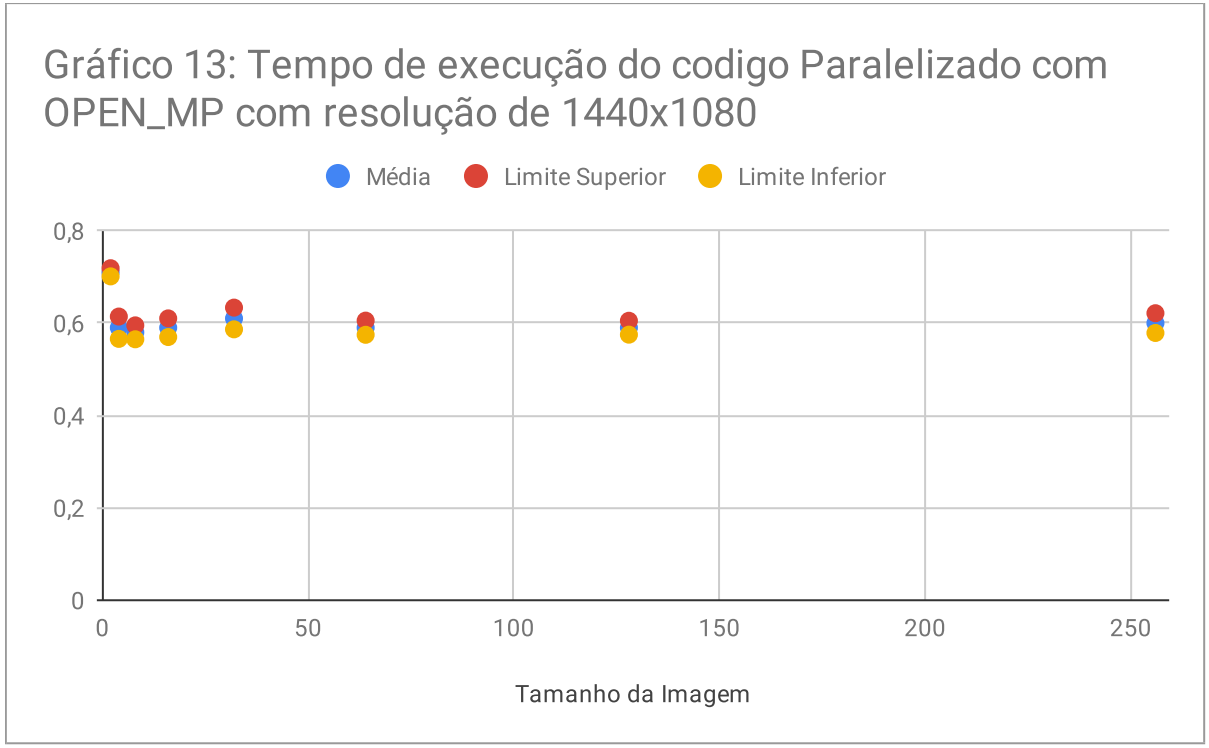

Tabela 14: Tempo de execução do codigo Paralelizado com OPEN_MP com resolução de 2048x1080							
Tamanho da Imagem	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	1,03	0,0486190243	1,078619024	0,9813809757	0,02460416897	1,01	1,045
4	0,94	0,08603985787	1,026039858	0,8539601421	0,04354137566	0,855	0,995
8	0,93	0,07433514839	1,004335148	0,8556648516	0,03761808423	0,895	0,985
16	0,92	0,04955035918	0,9695503592	0,8704496408	0,02507548079	0,885	0,93
32	0,89	0,01897366596	0,908973666	0,871026334	0,00960182336	0,88	0,89
64	0,9	0,02873524466	0,9287352447	0,8712647553	0,014541773	0,885	0,93
128	0,89	0,0324844285	0,9224844285	0,8575155715	0,01643908695	0,87	0,91
256	0,89	0,04700557211	0,9370055721	0,8429944279	0,02378766451	0,88	0,95

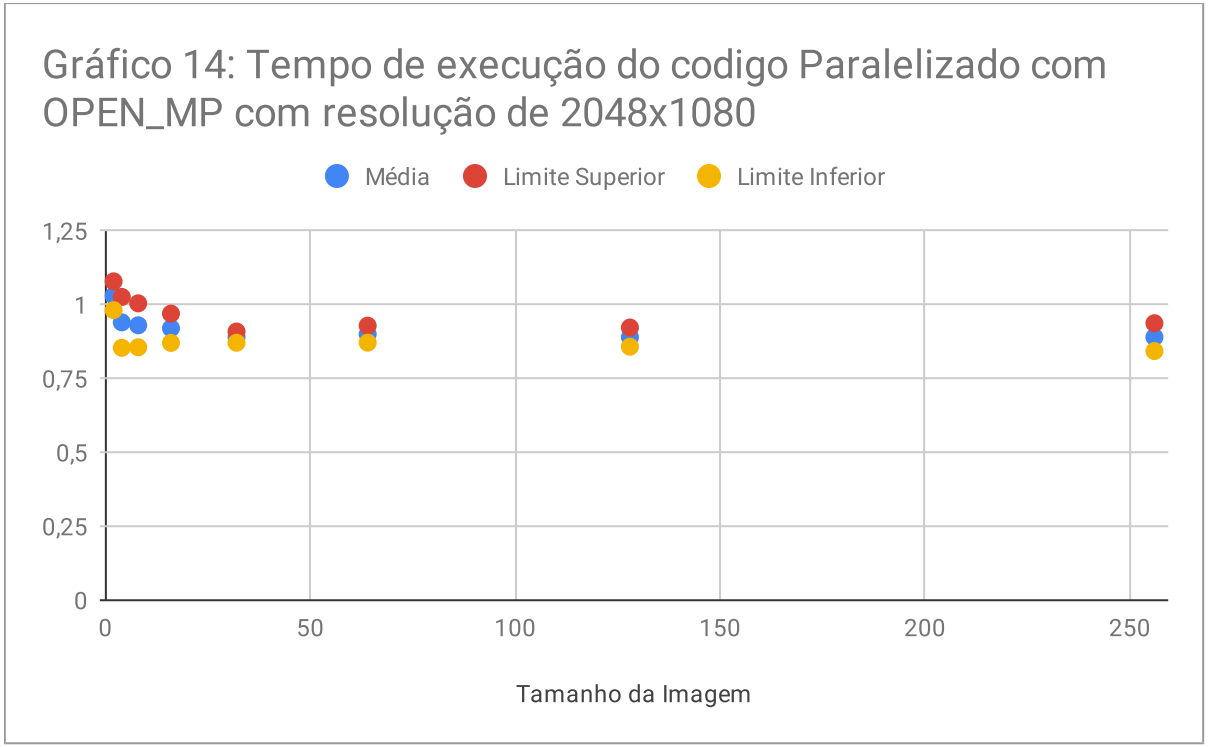

Tabela 15: Tempo de execução do codigo Paralelizado com OPEN_MP com resolução de 3840x2160							
Tamanho da Imagem	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	3,79	0,04945416348	3,839454163	3,740545837	0,02502679994	3,755	3,815
4	3,19	0,100985619	3,290985619	3,089014381	0,05110483537	3,1	3,235
8	3,15	0,1074687108	3,257468711	3,042531289	0,05438567225	3,12	3,25
16	3,15	0,08251983888	3,232519839	3,067480161	0,041760033	3,115	3,245
32	3,21	0,08373427358	3,293734274	3,126265726	0,04237461046	3,19	3,28
64	3,22	0,04876571786	3,268765718	3,171234282	0,02467840479	3,19	3,255
128	3,22	0,05897537824	3,278975378	3,161024622	0,02984511088	3,16	3,27
256	3,19	0,06776710819	3,257767108	3,122232892	0,03429425836	3,17	3,215

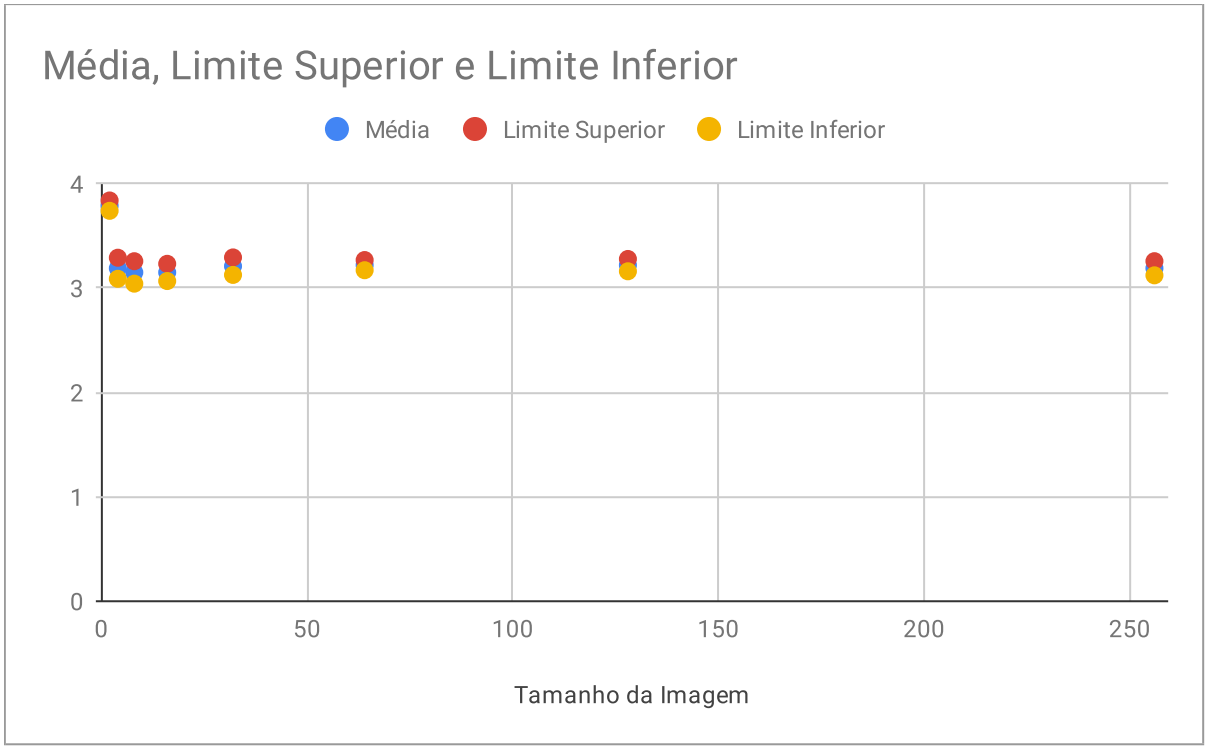

Tabela 16: Tempo de execução do codigo Paralelizado com OPEN_MP com rsolução de 7680x4320							
Tamanho da Imagem	**Média**	**Desvio Padrão**	**Limite Superior**	**Limite Inferior**	**Intervalo de Confiança**	**Primeiro Quartil**	**Terceiro Quatil**
2	15,16	0,6156738469	15,77567385	14,54432615	0,3115682303	15,045	15,97
4	13,98	1,279152621	15,25915262	12,70084738	0,6473286473	12,625	14,825
8	14,11	0,6943390142	14,80433901	13,41566099	0,3513775662	13,69	14,52
16	13,76	0,6468149069	14,40681491	13,11318509	0,3273274915	13,39	14,09
32	13,1	0,2124841731	13,31248417	12,88751583	0,1075298522	12,92	13,2
64	12,95	0,1806522574	13,13065226	12,76934774	0,09142097624	12,795	13,055
128	12,85	0,1609702723	13,01097027	12,68902973	0,08146070055	12,68	12,91
256	12,9	0,2017943319	13,10179433	12,69820567	0,1021201456	12,86	13,175