Parcial 1

Presentado por:

Alejandro Guerrero Cano: 202179652 – 3743

Presentado a:

John Sanabria

Asignatura:

Infraestructura paralelas y distribuidas

Universidad del Valle

Santiago de Cali

24 de octubre del 2024

Características de hardware

```
alejandro@Alejandro:/mnt/c/Users/alejo/OneDrive/Documents/Infraestructura/parcial1/paralelas-distribuidas-1er-parcial$ l
scpu
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         48 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  16
  On-line CPU(s) list:   0-15
Vendor ID:               AuthenticAMD
  Model name:            AMD Ryzen 7 7730U with Radeon Graphics
    CPU family:          25
    Model:               80
    Thread(s) per core:  2
    Core(s) per socket:  8
    Socket(s):           1
    Stepping:            0
    BogoMIPS:            3992.40
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse ss
                         e2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm constant_tsc rep_good nopl tsc_reliable nons
                         top_tsc cpuid extd_apicid pni pclmulqdq ssse3 fma cx16 sse4_1 sse4_2 movbe popcnt aes xsave avx
                          f16c rdrand hypervisor lahf_lm cmp_legacy svm cr8_legacy abm sse4a misalignsse 3dnowprefetch o
                         svw topoext perfctr_core ssbd ibrs ibpb stibp vmmcall fsgsbase bmi1 avx2 smep bmi2 erms invpcid
                          rdseed adx smap clflushopt clwb sha_ni xsaveopt xsavec xgetbv1 xsaves clzero xsaveerptr arat n
                         pt nrip_save tsc_scale vmcb_clean flushbyasid decodeassists pausefilter pfthreshold v_vmsave_vm
                         load umip vaes vpclmulqdq rdpid fsrm
Virtualization features:
  Virtualization:        AMD-V
  Hypervisor vendor:     Microsoft
  Virtualization type:   full
```

Tiempos:

Tiempo secuencial:

Tiempo secuencial	
12.844s	X
12.814s	
12.531s	X
12.608s	
12.641s	
Promedio:	12.687s

Tiempo paralelo núcleos 8:

Tiempo paralelo nucleos = 8	
12.670s	
12.654s	X
12.669s	
12.774s	X
12.667s	
Promedio:	12.668s

Tiempo paralelo núcleos 16:

Tiempo paralelo nucleos = 16	
12.181s	X
12.318s	
12.254s	
12.238s	
13.106s	X
Promedio:	12.270s

Speedups:

Speedup de paralelo núcleos = 8:

Tiempo secuencial	12.687s
Tiempo paralelo nucleos = 8	12.668s
Speedup	1.001s

Speedup de paralelo núcleos = 16:

Tiempo secuencial	12.687s
Tiempo paralelo nucleos = 16	12.270s
Speedup	1.033s

Con base a las tablas presentadas anteriormente, se observa una mejora entre un código en paralelo con 8 hilos y con 16 hilos, siendo el uso de 16 hilos más eficiente para el procesamiento de imágenes.

Sin embargo, en mi opinión, a pesar de la mejora con respecto a la versión secuencial, esta es mínima, especialmente en el caso de la ejecución con 8 hilos, donde la diferencia es prácticamente insignificante y manteniéndose constante, independientemente de las aplicaciones abiertas durante las pruebas (las mediciones se realizaron con el mínimo de aplicaciones abiertas o activas).

La explicación a esto es que el código no es debidamente concurrente y por ende no le sirve los procesos de paralelización con OMP.

Repositorio Github:

https://github.com/Alejo101102/parcial1-infraestructura