Relatório Trabalho 2

Programação Paralela

Victor Ribeiro Garcia (GRR20203954)
Álvaro R. S. Dziadzio (GRR20203913)

O Particionamento

O trabalho principal é realizado dentro de uma função, a função multi_partition particiona um vetor de entrada (input) em várias subpartições com base em valores de referência (part). O objetivo é reorganizar os elementos no vetor de saída (output), agrupando-os conforme as partições definidas, de forma eficiente e paralela, utilizando múltiplas threads.

Inicialmente, a função calcula quantos elementos de input pertencem a cada partição, armazenando esses valores no vetor auxiliar partition_block. Em seguida, calcula as posições iniciais de cada partição no vetor de saída (partition_starts) como uma soma acumulada das contagens. Um vetor atômico (partition_positions) é inicializado com essas posições para rastrear onde os elementos de cada partição serão escritos durante o processamento.

O trabalho é dividido entre as threads em "chunks", cada um processando uma fatia do vetor de entrada. As threads são inicializadas com dados relevantes, como o intervalo de elementos que devem processar e os vetores compartilhados. Uma barreira de sincronização (pthread_barrier_wait) garante que todas as threads iniciem o processamento simultaneamente.

Durante o processamento, cada thread determina a partição de cada elemento no seu intervalo usando busca binária no vetor part. Os elementos são então posicionados corretamente no vetor de saída, e as posições atuais de escrita são atualizadas de forma atômica, garantindo a segurança em um ambiente multithreaded.

Ao final, os recursos auxiliares (partition_block e partition_positions) são liberados, garantindo a limpeza da memória. A abordagem paralela da função permite particionar grandes vetores de forma eficiente, explorando o poder das threads e minimizando condições de corrida.

Implementação

Funções importantes:

- multi_partition: para a partição do vetor de entrada (input).
    - Calcula o tamanho de cada partição e inicializa os índices de início;
    - Divide o trabalho em blocos, um para cada thread;

    - As threads identificam a partição de cada elemento usando busca binária e distribuem os elementos no vetor de saída (output).

- verifica_particoes: verificar a correção do particionamento.
    - Executado após multi_partition;
    - Percorre cada partição e valida se os valores estão dentro dos limites esperados (lower_bound e upper_bound).

Funções auxiliares para pool de threads:

- Thread_pool_create (inicializa um pool de threads) e stop_thread_pool (finaliza as threads do pool).
    - Utiliza uma barreira para sincronizar todas as threads antes de iniciar ou finalizar uma tarefa.

Houveram alterações no script roda-todos-slurm.sh, agora o primeiro parâmetro passado para roda-todos-slurm.sh deve ser o nome do executável.

O makefile foi feito para gerar 2 executáveis (parteA_1k para parte A do trabalho atribuindo N_PART=1000 e parteB-100K para a parte B do trabalho atribuindo N_PART=100000).

A execução deve ser feita da seguinte forma:
1) Make
2) sbatch –exclusive roda-todos-slurm.sh parteA_1k (para 1000 elementos no vetor de particionamento) ou sbatch –exclusive roda-todos-slurm.sh parteB_100k (para 100mil elementos no vetor de particionamento).

Além disso, foi feito um script adicional gera_resultados.py (não estava na especificação) para gerar gráficos .png e resumo das estatísticas de tempo médio em segundos e vazão média em MEPS (milhões de elementos particionados por segundo).

Observação: as planilhas com os mesmos cálculos também foram feitas no formato .ods (estão no diretório planilhas), o script foi apenas um adicional.

O Processador Usado

O processador possui uma configuração composta por dois pacotes físicos (sockets), identificados como Package L#0 e Package L#1, cada um contendo quatro núcleos físicos. No total, há oito núcleos físicos disponíveis no sistema. Cada núcleo é associado a um único núcleo lógico (PU - Processing Unit). Os núcleos dentro de cada pacote compartilham um cache L2 de 6 MB, organizado em grupos de dois núcleos por cache.

lscpu

```
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   38 bits physical, 48 bits virtual
CPU(s):                          8
On-line CPU(s) list:             0-7
Thread(s) per core:              1
Core(s) per socket:              4
Socket(s):                       2
NUMA node(s):                    1
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           23
Model name:                      Intel(R) Xeon(R) CPU
E5462  @ 2.80GHz
Stepping:                        6
CPU MHz:                         2793.098
BogoMIPS:                        5585.67
Virtualization:                  VT-x
L1d cache:                       256 KiB
L1i cache:                       256 KiB
L2 cache:                        24 MiB
NUMA node0 CPU(s):               0-7
Vulnerability Itlb multihit:     KVM: Mitigation: VMX disabled
Vulnerability L1tf:              Mitigation; PTE Inversion; VMX EPT
disabled
Vulnerability Mds:               Vulnerable: Clear CPU buffers
attempted, no microcode; SMT disabled
Vulnerability Meltdown:          Mitigation; PTI
Vulnerability Spec store bypass: Vulnerable
Vulnerability Spectre v1:        Mitigation; usercopy/swapgs
barriers and __user pointer sanitization
Vulnerability Spectre v2:        Mitigation; Full generic retpoline,
STIBP disabled, RSB filling
Vulnerability Srbds:             Not affected
Vulnerability Tsx async abort:   Not affected
Flags:                           fpu vme de pse tsc msr pae mce cx8
apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ht tm pbe syscall nx lm constant_tsc arch_perfmon pebs bts
rep_good nopl cpuid aperfmperf pni dtes64 monitor ds_cpl vmx est tm2
```

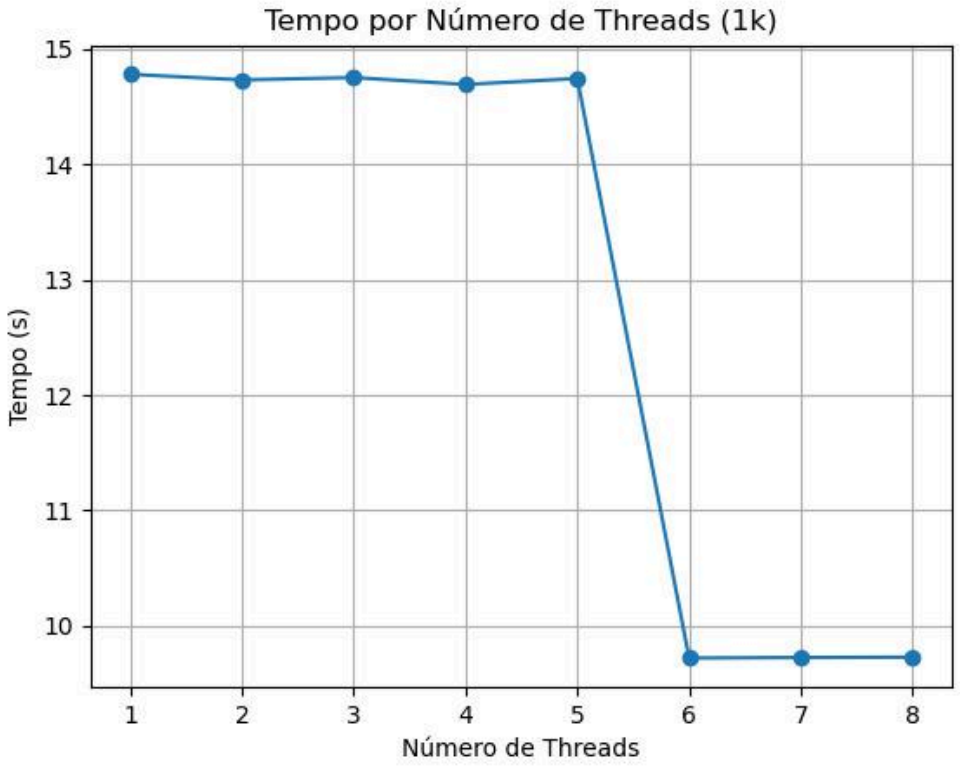
```
ssse3 cx16 xtpr pdcm dca sse4_1 lahf_lm pti tpr_shadow vnmi
flexpriority vpid dtherm
```

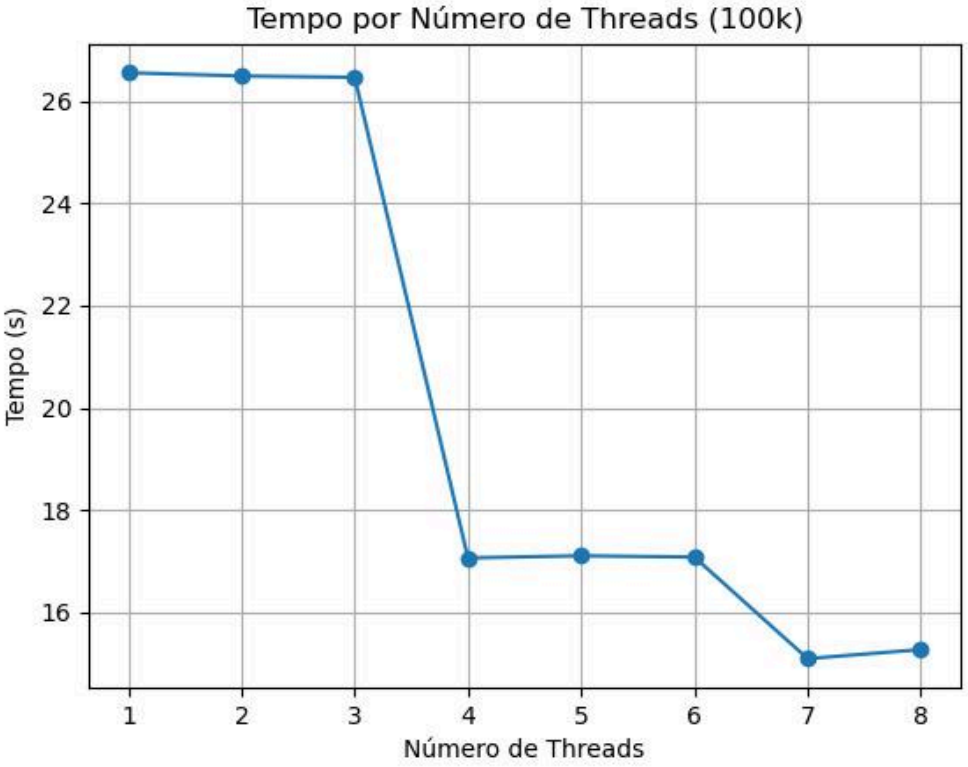
lstopo

```
Machine (31GB total)
  NUMANode L#0 (P#0 31GB)
  Package L#0
    L2 L#0 (6144KB)
      L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0 + PU L#0 (P#0)
      L1d L#1 (32KB) + L1i L#1 (32KB) + Core L#1 + PU L#1 (P#2)
    L2 L#1 (6144KB)
      L1d L#2 (32KB) + L1i L#2 (32KB) + Core L#2 + PU L#2 (P#4)
      L1d L#3 (32KB) + L1i L#3 (32KB) + Core L#3 + PU L#3 (P#6)
  Package L#1
    L2 L#2 (6144KB)
      L1d L#4 (32KB) + L1i L#4 (32KB) + Core L#4 + PU L#4 (P#1)
      L1d L#5 (32KB) + L1i L#5 (32KB) + Core L#5 + PU L#5 (P#3)
    L2 L#3 (6144KB)
      L1d L#6 (32KB) + L1i L#6 (32KB) + Core L#6 + PU L#6 (P#5)
      L1d L#7 (32KB) + L1i L#7 (32KB) + Core L#7 + PU L#7 (P#7)
  HostBridge
    PCIBridge
      PCI 01:00.0 (SCSI)
        Block "sda"
    PCIBridge
      PCIBridge
        PCIBridge
          2 x { PCI 05:00.0-1 (Ethernet) }
    PCIBridge
      PCI 08:01.0 (VGA)
    PCI 00:1f.2 (RAID)
  Block "sdb"
  Block "sdc"
```

Resultados

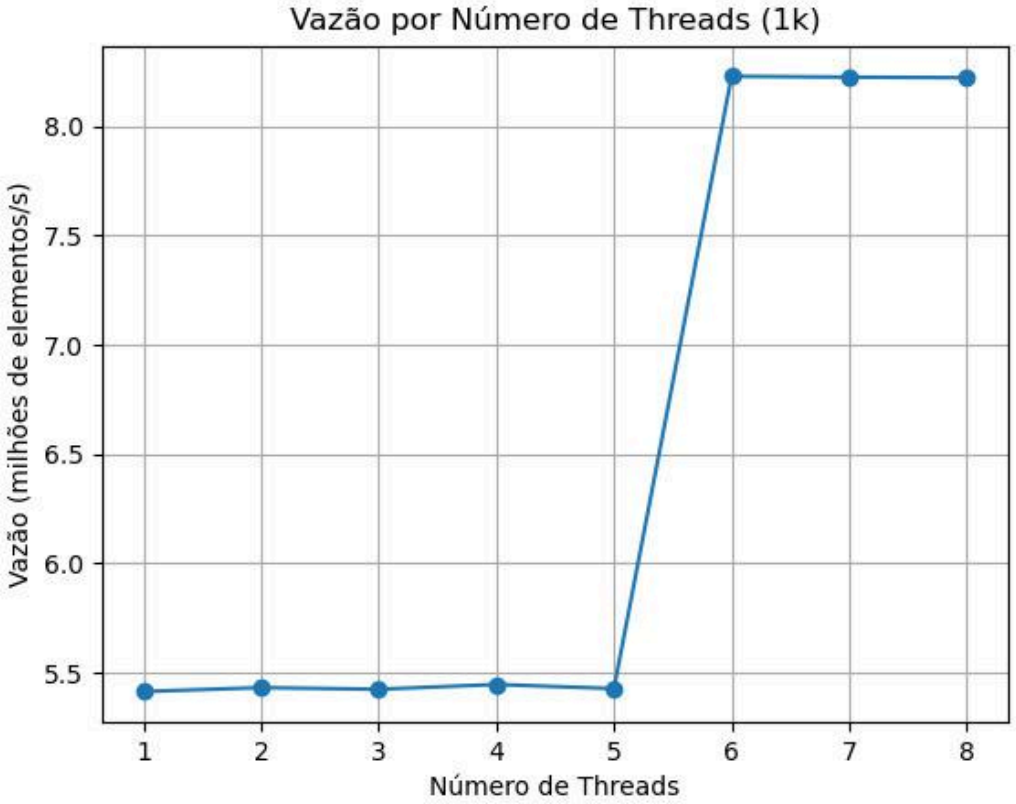
Parte A) Para 1000 elementos, as médias foram:
1 threads - Tempo: 14.779407s, Vazão: 5.41M elem/s
2 threads - Tempo: 14.730487s, Vazão: 5.43M elem/s
3 threads - Tempo: 14.751037s, Vazão: 5.42M elem/s
4 threads - Tempo: 14.691433s, Vazão: 5.45M elem/s
5 threads - Tempo: 14.742737s, Vazão: 5.43M elem/s
6 threads - Tempo: 9.720989s, Vazão: 8.23M elem/s
7 threads - Tempo: 9.725860s, Vazão: 8.23M elem/s
8 threads - Tempo: 9.728791s, Vazão: 8.22M elem/s

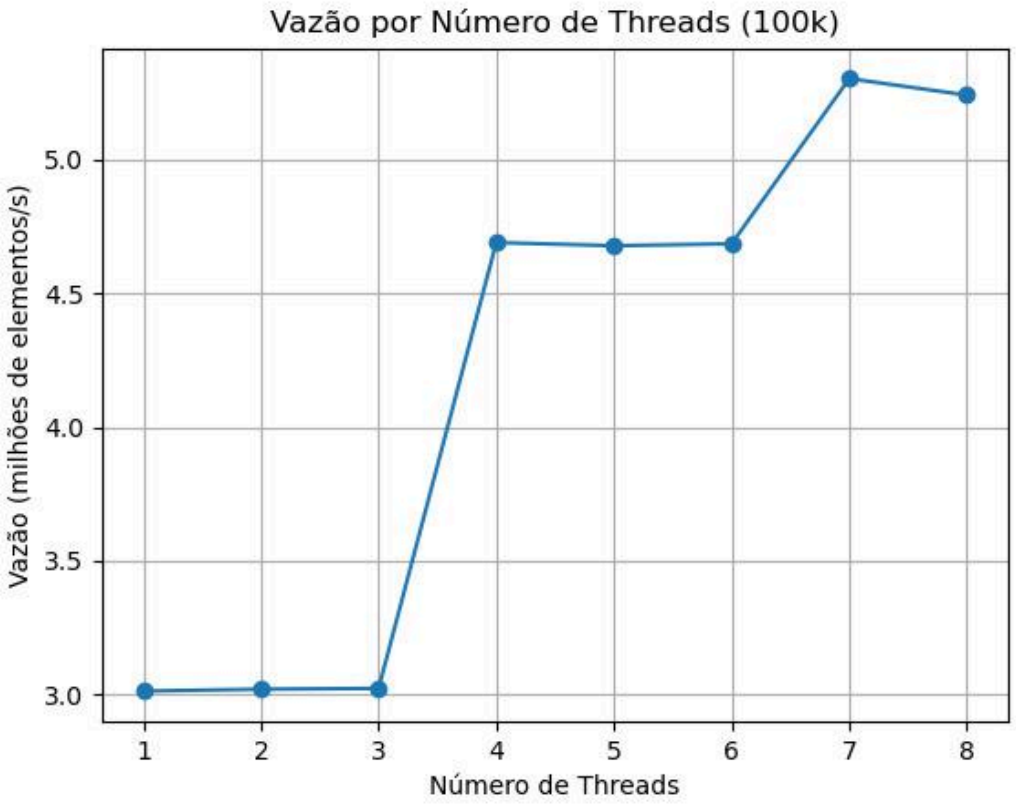




Parte B) Para 100000 elementos, as médias foram:

1 threads - Tempo: 26.562777s, Vazão: 3.01M elem/s
2 threads - Tempo: 26.499548s, Vazão: 3.02M elem/s
3 threads - Tempo: 26.473263s, Vazão: 3.02M elem/s
4 threads - Tempo: 17.057144s, Vazão: 4.69M elem/s
5 threads - Tempo: 17.099714s, Vazão: 4.68M elem/s
6 threads - Tempo: 17.074065s, Vazão: 4.69M elem/s
7 threads - Tempo: 15.084311s, Vazão: 5.30M elem/s
8 threads - Tempo: 15.261869s, Vazão: 5.24M elem/s


Vazão por Número de Threads (1k)
Vazão (milhões de elementos/s)
8.0
7.5
7.0
6.5
6.0
5.5
1
2
3
4
5
6
7
8
Número de Threads


Vazão por Número de Threads (100k)
Vazão (milhões de elementos/s)
5.0
4.5
4.0
3.5
3.0
1
2
3
4
5
6
7
8
Número de Threads