Relatório Trabalho 3

Programação Paralela

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

O Particionamento

A função multi_partition_mpi realiza a partição e redistribuição de elementos de um array em um ambiente de computação paralela utilizando MPI (Message Passing Interface). Ela divide os elementos de entrada (input) em partições definidas por pivôs (P) e redistribui os dados entre processos MPI para que cada processo receba os elementos correspondentes às partições que lhe pertencem.

Inicialmente, a função conta quantos elementos de input pertencem a cada partição, utilizando busca binária nos pivôs. Com essas informações, calcula as posições iniciais para inserção dos elementos no array output, onde são organizados localmente por partição. Em seguida, define os tamanhos e as posições dos dados que serão enviados para outros processos.

Para redistribuir os elementos, a função usa MPI_Alltoall para compartilhar as informações de tamanho entre os processos e calcular as posições de recebimento no buffer local. Os dados particionados são então transferidos utilizando MPI_Alltoallv, garantindo que cada processo receba os elementos pertencentes às partições atribuídas a ele.

Por fim, a função libera a memória dos vetores auxiliares alocados. Após sua execução, os elementos redistribuídos ficam disponíveis no buffer local de cada processo, e o total de elementos recebidos é armazenado em partitionStart.

Implementação

O MPI é usado para coordenar a execução paralela, distribuir dados (como pivôs) de forma eficiente e redistribuir os elementos particionados entre os processos. Ele garante sincronização entre os processos e permite que o programa escale para processar grandes volumes de dados em sistemas distribuídos.

O programa espera receber de entrada o tamanho do vetor de entrada com 8 milhões de elementos por padrão e o número de processos n que serão utilizados no MPI (./multipart-mpi 8000000 n). Dentro do programa principal, a função multi_partition_mpi é executada e medida NTIMES vezes (NTIMES = 10). Assim, métricas como tempo total de execução em segundo e vazão são medidas 10 vezes.

Além disso diversas funções foram reutilizadas do Trabalho 2 como: alocaVetInt (alocar vetor de inteiros), alocaVetLongLong (alocar vetor do tipo long long), randomLongLong (gera um número long long aleatório), randomVet (preenche um vetor com valores aleatórios), compara (função de comparação para qsort) e busca_binaria (encontrar a partição correta).

Funções importantes

- **`multi_partition_mpi`**: para a partição do vetor de entrada (input).
    - Aloca vetores auxiliares
    - Para cada elemento do array de entrada, utiliza a função busca_binaria para determinar a qual partição ele pertence, incrementando o contador correspondente em num_part
    - Baseado nas contagens de elementos em cada partição (num_part), calcula as posições iniciais
    - Utiliza MPI_Alltoallv para redistribuir os elementos particionados.

- **`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).

Execução

Para a execução do programa em um ambiente gerenciado pelo SLURM, foram utilizados quatro scripts distintos para testar diferentes configurações de distribuição de processos MPI. Cada script especifica o número de nós computacionais (--nodes, denotado por n) e o número de tarefas por nó (--ntasks-per-node, denotado por ppn). Os scripts rodar-exp1-slurm-1n-8ppn, rodar-exp2-slurm-2n-8ppn e rodar-exp3-slurm-4n-8ppn configuram 8 processos por nó em 1, 2 e 4 nós, respectivamente, enquanto o script rodar-slurm-1n-1ppn executa apenas 1 processo em um único nó.

Para utilizar o script, deve-se usar o comando sbatch nome_do_script.sh. Ele aloca os recursos especificados, como o número de nós e tarefas por nó, executa o programa MPI com os parâmetros definidos e imprime informações sobre o job, incluindo o tempo total de execução e detalhes do ambiente SLURM.

A execução deve ser feita da seguinte forma:

- *make*
- Experimento Base (1 nodo e 1 processo por nodo):
    - *sbatch --exclusive ./rodar-exp0-slurm-1n-1ppn.sh*
- Experimento 1 (1 nodo e 8 processos por nodo):
    - *sbatch --exclusive ./rodar-exp1-slurm-1n-8ppn.sh*

- Experimento 2 (2 nodos e 8 processos por nodo):
    - *sbatch --exclusive ./rodar-exp2-slurm-2n-8ppn.sh*
- Experimento 3 (4 nodos e 8 processos por nodo):
    - *sbatch --exclusive ./rodar-exp3-slurm-4n-8ppn.sh*

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:

	1n-1ppn:	1n-8ppn:	2n-8ppn:	4n-8ppn:		
	0.321147	0.162119	0.266284	0.201299		
	0.276969	0.142805	0.247682	0.190019	total elements:	8000000
	0.275924	0.133422	0.263330	0.269945		
	0.275890	0.133886	0.292711	0.203193	NTIMES:	10
	0.277026	0.127694	0.244126	0.200376		
	0.275819	0.130893	0.243686	0.188018		
	0.277868	0.128261	0.248048	0.196111		
	0.278151	0.126910	0.250201	0.187804		
	0.278394	0.126776	0.262502	0.183972		
	0.278644	0.127180	0.240264	0.186833		

	Tempo Médio			
nodos/processos	1n-1ppn	1n-8ppn	2n-8ppn	4n-8ppn
tempo médio (s)	0,28	0,13	0,23	0,2
Aceleracao	1.00	2.15	1,22	1,40

	Vazão Média			
nodos/processos	1n-1ppn	1n-8ppn	2n-8ppn	4n-8ppn
vazao média (GFLOPS)	0,286	0,615	0,348	0,4
Aceleracao	1.00	2.15	1.22	1.40