Relatório Programação Paralela Trabalho 1

Este programa em C foi desenvolvido por Gabriel Pimentel Dolzan e Túlio de Pádua Dutra para encontrar os "k" menores elementos de um grande conjunto de dados utilizando múltiplas threads para acelerar o processo.

Estruturas de Dados

- *pair_t*: Estrutura que armazena um par de chave e valor, onde `key` é um float e `val` é um inteiro.
- *thread_data_t*: Estrutura utilizada para armazenar informações específicas para cada thread.

Funções Auxiliares

- swap(): Troca os valores de dois elementos.
- compare(), compare_verify() e compare_pair(): São funções de comparação para qsort(). Cada uma utilizada em um certo contexto, para comparar float, pair_t, etc.
- maxHeapify() e heapifyUp(): São funções de ajuste para uma estrutura de max heap.
- insert(): Insere um elemento na estrutura de max heap.
- decreaseMax(): Substitui a raiz (máximo) da heap se o novo valor for menor.
- fillArrayRandom(): Preenche o array Input e InputPair com números aleatórios.
- binary_search(): Busca binária em um array de `pair_t` para verificar se um par específico existe.
- verifyOutput(): Verifica se os k menores elementos no array `Output` são corretos.

3. **Thread Function**

- thread_function(): Função que será executada por cada thread. Ela processa uma partição do conjunto de dados e preenche uma heap local com os k menores elementos da partição.

4. **Merge Function**

- merge_function(): Função que será executada por cada thread. Ela unifica as partições do conjunto de dados e preenche uma heap com os k menores elementos totais.

5. **Encontrando os k Menores Elementos**
- findKSmallest(): Utiliza múltiplas threads para processar diferentes partições do conjunto de dados. Cada thread preenche uma heap local e, depois, todas as heaps são mescladas para encontrar os k menores elementos globais.

6. **Main**
- Primeiro, verifica os argumentos de entrada.
- Inicializa os arrays e preenche o array `Input` com valores aleatórios.
- Executa a função findKSmallest() e mede o tempo necessário para isso.
- Verifica a saída usando verifyOutput().
- Libera a memória alocada.

Fluxo de Execução
1. O programa começa lendo três argumentos da linha de comando: o número total de elementos, k (o número de menores elementos desejados) e o número de threads a serem usados.
2. Depois, ele aloca memória para os arrays Input, InputPair e Output e preenche Input com números aleatórios.
3. A função findKSmallest() é chamada para encontrar os k menores elementos.
4. Os k menores elementos encontrados são armazenados no array Output.
5. Finalmente, o programa verifica se os k menores elementos no array Output são corretos e exibe uma mensagem indicando se a saída é válida ou não.

Observação
Esse programa foi projetado para usar um número específico de threads (entre 1 e 8). Ele divide o conjunto de dados em partições, e cada thread processa uma partição. A ideia principal é que cada thread identifica os k menores elementos em sua partição e, em seguida, todos

esses k menores elementos locais são combinados para obter os k menores elementos globais.

Como foi realizado o experimento

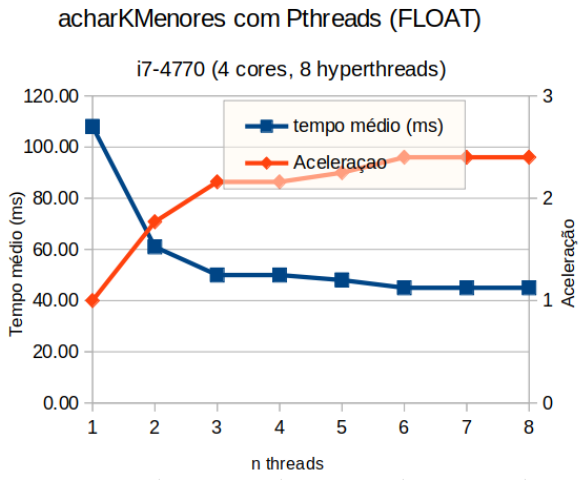
O programa foi rodado 10 vezes com 1, 2, 3, 4, 5, 6, 7 e 8 threads. Totalizando 80 execuções de programa. O tamanho do vetor escolhido é 100000000. E o tamanho do K é 2048.

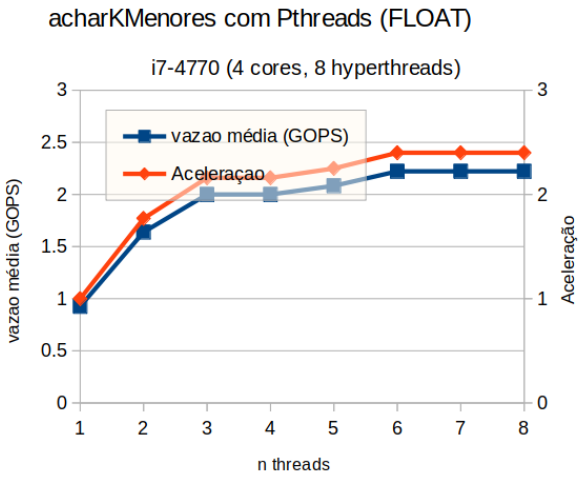
Dados da planilha

1 threads:	2 threads:	3 threads:	4 threads:	5 threads:	6 threads:	7 threads:	8 threads:
0.121370	0.059182	0.050418	0.050418	0.045916	0.044379	0.046472	0.042572
0.096452	0.060860	0.045225	0.045225	0.049967	0.044941	0.045452	0.043162
0.106820	0.066188	0.050223	0.050223	0.046845	0.045066	0.046353	0.043642
0.119856	0.055560	0.045727	0.045727	0.045978	0.045053	0.043751	0.043314
0.121195	0.062400	0.050263	0.050263	0.046549	0.042815	0.046030	0.043954
0.106555	0.065354	0.080805	0.080805	0.046590	0.043803	0.043711	0.049259
0.106788	0.058212	0.044067	0.044067	0.046524	0.052170	0.043715	0.043140
0.096727	0.066821	0.045043	0.045043	0.049917	0.045393	0.043853	0.050593
0.096679	0.055980	0.045016	0.045016	0.054545	0.044445	0.043805	0.042636
0.107593	0.056427	0.047923	0.047923	0.049016	0.045659	0.046356	0.048765

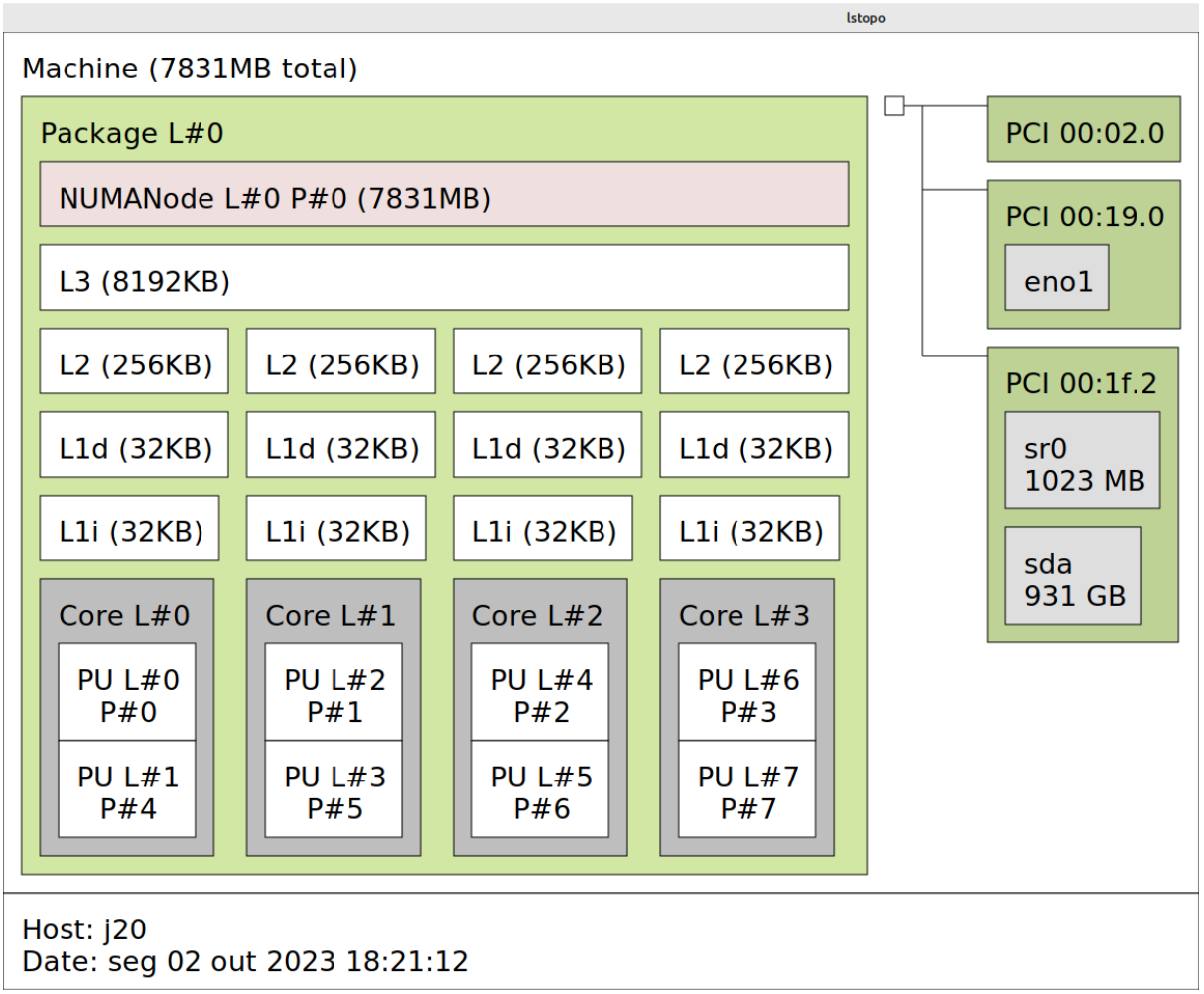
	Tempo médio							
threads	1	2	3	4	5	6	7	8
tempo médio (ms)	108.00	61.00	50	50	48	45	45	45
Aceleraçao	1.00	1.77	2.16	2.16	2.25	2.40	2.40	2.40

	VAZAO MEDIA:							
threads	1	2	3	4	5	6	7	8
vazao média (GOPS)	0.926	1.639	2	2	2.083	2.222	2.222	2.222
Aceleraçao	1.00	1.77	2.16	2.16	2.25	2.40	2.40	2.40





Comando LSTOPO



Comando LSCPU

Arquitetura: x86_64
Modo(s) operacional da CPU: 32-bit, 64-bit
Ordem dos bytes: Little Endian
Tamanhos de endereço: 39 bits physical, 48 bits virtual
CPU(s): 8
Lista de CPU(s) on-line: 0-7
Thread(s) per núcleo: 2
Núcleo(s) por soquete: 4
Soquete(s): 1
Nó(s) de NUMA: 1
ID de fornecedor: GenuineIntel
Família da CPU: 6
Modelo: 60
Nome do modelo: Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz

Step: 3
CPU MHz: 844.593
CPU MHz máx.: 3900,0000
CPU MHz mín.: 800,0000
BogoMIPS: 6784.95
cache de L1d: 128 KiB
cache de L1i: 128 KiB
cache de L2: 1 MiB
cache de L3: 8 MiB
CPU(s) de nó0 NUMA: 0-7
Vulnerability Itlb multihit: KVM: Mitigation: VMX unsupported
Vulnerability L1tf: Mitigation; PTE Inversion
Vulnerability Mds: Vulnerable: Clear CPU buffers attempted, no mic
rocode; SMT vulnerable
Vulnerability Meltdown: Mitigation; PTI
Vulnerability Mmio stale data: Unknown: No mitigations
Vulnerability Retbleed: Not affected
Vulnerability Spec store bypass: Vulnerable
Vulnerability Spectre v1: Mitigation; usercopy/swapgs barriers and __user
pointer sanitization
Vulnerability Spectre v2: Mitigation; Retpolines, STIBP disabled, RSB fil
ling, PBRSB-eIBRS Not affected
Vulnerability Srbds: Vulnerable: No microcode
Vulnerability Tsx async abort: Not affected
Opções: fpu vme de pse tsc msr pae mce cx8 apic sep mtr
r pge mca cmov pat pse36 clflush dts acpi mmx f
xsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rd
tscp lm constant_tsc arch_perfmon pebs bts rep_
good nopl xtopology nonstop_tsc cpuid aperfmper
f pni pclmulqdq dtes64 monitor ds_cpl smx est t
m2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 ss

e4_2 x2apic movbe popcnt aes xsave avx f16c
rdr

and lahf_lm abm cpuid_fault epb invpcid_single
pti fsgsbase tsc_adjust bmi1 avx2 smep bmi2
erm

s invpcid xsaveopt dtherm ida arat pln pts