Relatório T1 - Programação Paralela (CI1316)

Alexandre de Oliveira Plugge Freitas Wilian Pereira dos Santos
Universidade Federal do Paraná

3 de outubro de 2023

Resumo

Trabalho da disciplina de Programação paralela que utiliza um determinado número de threads para o problema: Achar os K menores valores dado uma lista com N números reais (Floats).

1 Introdução

A implementação adotada neste trabalho envolve a exploração da paralelização, em CPUs multicore usando a biblioteca PThreads para resolver um problema computacional.

2 O problema

O problema computacional a ser resolvido no trabalho é o de achar os K menores valores em uma entrada de N valores Float, tal que K é menor ou igual a N. É também um requisito do trabalho que se use um numero de threads dado como entrada no programa.

3 O programa

A entrada do programa é composta de 3 valores:

- N: Número total de Floats
- K: O número de elementos que queremos achar dentro dos N Floats
- numThreads: O número de threads que serão usadas na execução paralela.

3.1 DecreaseMax

A estratégia usada para achar os K menores elementos em uma entrada com K números reais é o algoritmo DecreaseMax utilizando a estrutura de dados heap (implícita). Nesse algoritmo, é criada um max-heap com K elementos (nesse caso, os K primeiros elementos da entrada), depois disso, cada elemento da entrada é inserido na heap utilizando o DecreaseMax. Caso o número a ser inserido seja maior ou igual que a cabeça da heap, ele não é inserido, se for menor, é inserido no lugar da cabeça e a heap é ajustada para voltar a ser um max-heap.

Mantendo sempre o tamanho da heap igual a K, sempre serão encontrados os K menores valores dentro de um intervalo de números maior que K, ideia essa que será usada para paralelizar a execução.

3.2 Paralelismo

Caso o número de threads (numThreads) seja 1, a thread 0 será naturalmente responsável por executar o algoritmo acima usando uma heap e a entrada inteira, ou seja, do elemento 0 até o N-1-ésimo elemento. Caso numThreads seja maior que 1, aloca-se memória para numThreads heaps de tamanho K (máximo), de modo que cada thread opere em cima de apenas uma heap. Além disso, a entrada é dividida em numThreads intervalos, para que cada thread opere sobre apenas esse intervalo, de modo

que não seja necessário tratar condições de disputa (Race conditions). Assim, cada heap, ao final da execução da thread, terá os K menores valores do seu intervalo, ou seja, são encontrados os K * numThreads elementos da entrada quando todas as threads finalizarem.

Por fim, todos os elementos de todas as heaps são inseridas em uma nova heap de K elementos utilizando o DecreaseMax, assim encontrando os K menores elementos dentre os K * numThreads elementos encontrados pelas threads, portanto, são também os K menores entre os N elementos da entrada.

4 Resultados

Os resultados a seguir foram obtidos rodando o programa 8 vezes, iniciando com 1 thread (sequencial) e aumentando o número de threads em 1 unidade a cada execução. O numero de floats N foi 100000000 (cem milhões) e K igual a 2048 em todas as execuções.

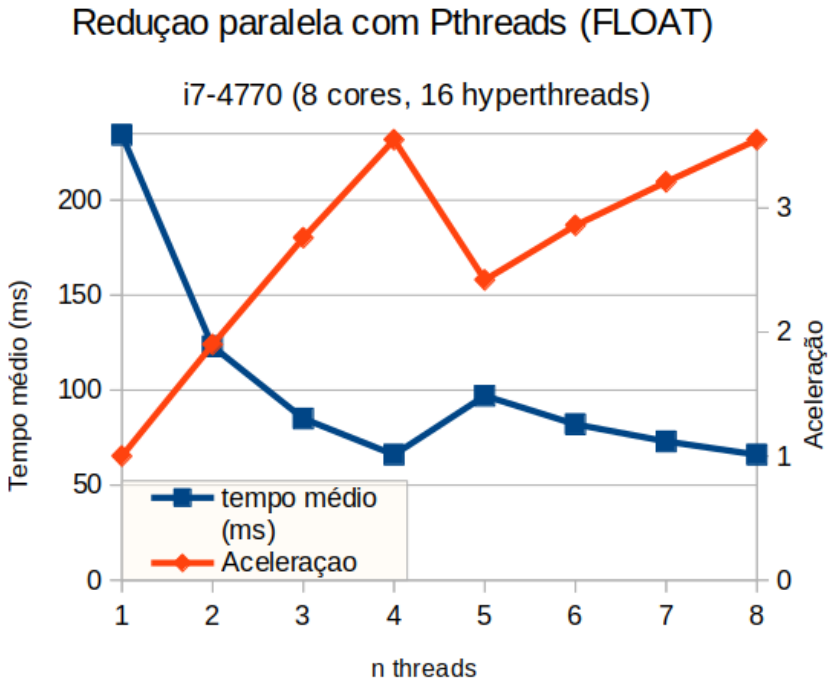

Figura 1: Tempo de execução e aceleração do algoritmo por número de threads utilizadas

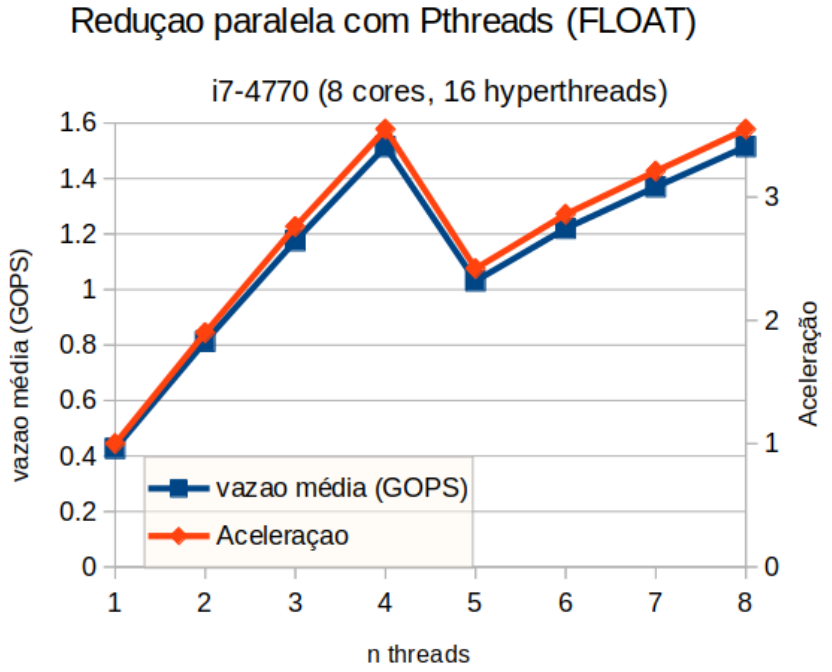

Figura 2: Vazão média por número de threads utilizadas

5 Hardware

Resultados do lscpu e lstopo no host J6 do departamento de informática da Universidade Federal do Paraná:

- **Arquitetura:** x86_64
- **Modo(s) operacional da CPU:** 32-bit, 64-bit
- **Ordem dos bytes:** Little Endian
- **Tamanhos de endereço:** 39 bits físicos, 48 bits virtuais
- **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:** 800.000
- **CPU MHz máx.:** 3900,0000
- **CPU MHz mín.:** 800,0000
- **BogoMIPS:** 6783.77
- **Virtualização:** VT-x
- **Cache de L1d:** 128 KiB
- **Cache de L1i:** 128 KiB
- **Cache de L2:** 1 MiB
- **Cache de L3:** 8 MiB

Resultado do lstopo no host J6:

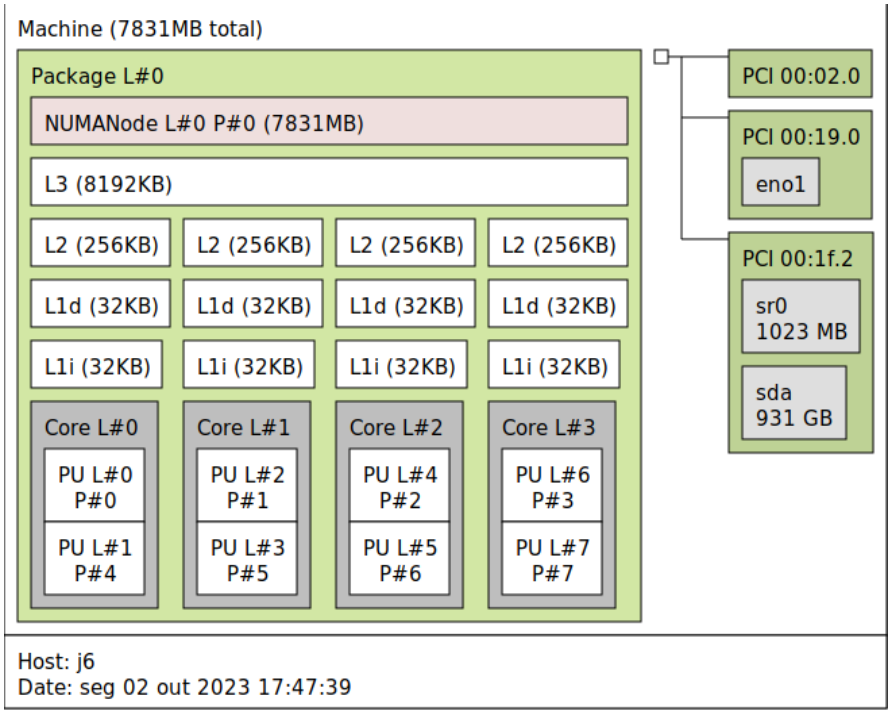

Figura 3: lstopo

6 Conclusão

Observa-se uma melhoria substancial no desempenho do programa à medida que o número de threads aumenta. A paralelização do processamento desse grande volume de dados resulta em uma redução significativa no tempo de execução. Isso sugere que a estratégia de distribuir a carga de trabalho entre várias threads é eficaz em acelerar o processamento.

Entretanto, é importante notar que a relação entre o número de threads e o tempo de execução não é estritamente linear. Embora mais threads geralmente levem a tempos de execução menores, a taxa de redução do tempo de execução diminui na quinta thread. Isso indica que pode haver limitações de escalabilidade, onde a introdução de mais threads não resulta em melhorias significativas e pode até mesmo aumentar a sobrecarga do sistema.

A escolha do número ideal de threads a serem utilizadas é, portanto, uma decisão importante e depende de vários fatores, incluindo as características específicas da CPU, o tamanho dos dados de entrada e a complexidade do algoritmo. É essencial encontrar um equilíbrio entre o desempenho desejado e o consumo de recursos, uma vez que o uso excessivo de threads pode resultar em ineficiência.