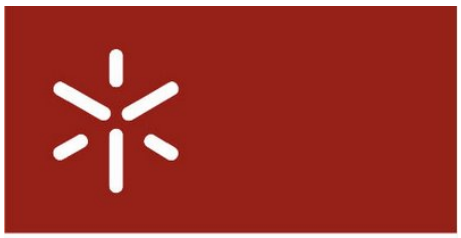
Universidade do Minho

Departamento de Informática

Algoritmos Paralelos

Escalabilidade de diferentes algoritmos paralelos



Universidade do Minho

António Sérgio Alves Costa A78296
José Pedro Moreira Resende A77486

1 de Junho de 2019

Conteúdo

1 Introdução **2**

2 Algortimo: Heapsort **2**
2.1 Sequencial 2
2.2 Paralelo 2

3 Apresentação e Análise dos Resultados **3**
3.1 Apresentação dos Resultados 3
3.2 Tempos de execuçao 3
3.2.1 Sequencial 3
3.2.2 Paralelo 3
3.3 Análise dos Resultados 3

4 Conclusão **3**

1 Introdução

Algoritmos de ordenação sao usados para reorganizar e ordenar uma dada lista ou array de acordo com parametros fornecidos por um comparador de elementos.

Existem varios algoritmos de ordenação com diferentes perfis de execucação e memoria, neste artigos vamos falar em especifico do heapsort ,da sua paralelização e da sua caracterização.

O heapsort é um algoritmo de ordenação da familia dos algoritmos de ordenacao por seleção.

Este algoritmo usa uma estrutura de dados heap para os ordenar a medida que os insere de forma a tornar a ordenacao estavel.

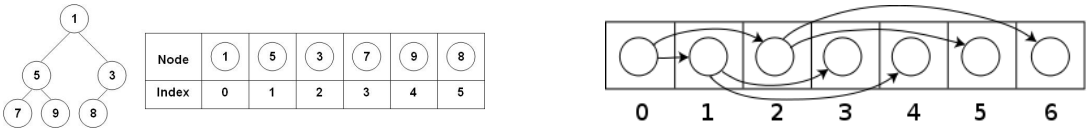
O seu tempo de execução sobre conjuntos ordenados aleatoriamente é muito bom, tem um uso de memória bem comportado e o seu desempenho em pior caso é praticamente igual ao desempenho em caso médio.

2 Algortimo: Heapsort

2.1 Sequencial

Nesta implementação do algoritmo heap sort sequencial utilizamos o codigo disponibilizado no enunciado.

O funcionamento deste algoritmo consiste em duas fases. Primeiramente e construido uma heap apartir do array que nos e dado, em segundo é construido o array ordenado retirando o maior numero (root atual) e inserindo no array ordenado. A heap e entao atualizada de forma a manter os invariantes deste tipo de estrutura e eleger um novo root.



Este array ordenado e na verdade feito no mesmo array da heap e simplesmente e alterado os limites final da heap, reaproveitando a memoria, sem ser necessario alocar um novo array para guardar o resultado final.

2.2 Paralelo

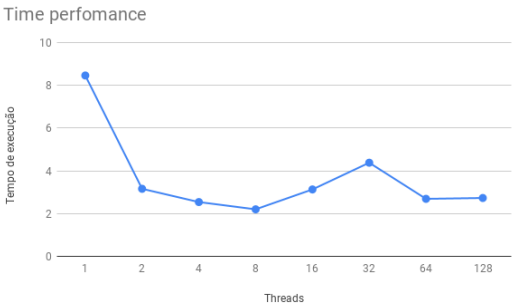
Com o intuito de paralelizar e acelarar este algoritmo depois de alguma pesquisa e experimentação chegamos a uma implementação na qual dividimos o array em varias hheaps mais pequenas de tamanho constante (sempre que possivel).

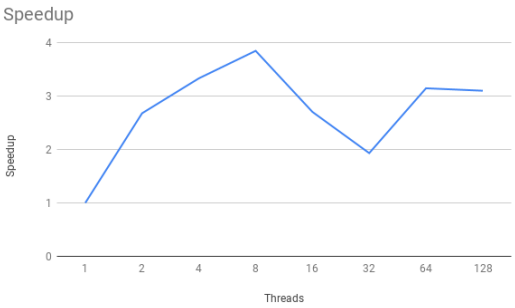
Assim tanto as heaps como os arrays ordenados associados a cada uma destas estruturas podem ser calculados e ordenados de forma independente e paralela.

Utilizamos aqui o scheduler dinamico do omp de forma a distribuir melhor a carga de trabalho entre as threads.

Apos obtermos varios segmentos ordenados e necessario correu o processo de fundir estes segmenetos de forma a obter um unico array ordenado.

Para este trabalho foi usado uma heuristica de redução em arvore o que torna todo o processo de merge extremamente efeciente e possivel de executar de forma paralela, uma vez que temos blocos sem dependencias entre si, tirando partido tambem de um scheduler dinamico de forma a que a carga de trabalho seja distribuida pelas threads.





3 Apresentação e Análise dos Resultados

3.1 Apresentação dos Resultados

3.2 Tempos de execuçao

3.2.1 Sequencial

3.2.2 Paralelo

3.3 Análise dos Resultados

4 Conclusão