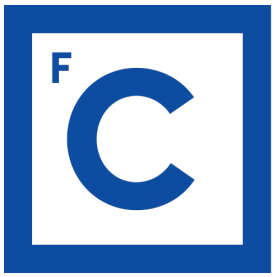

Ciências ULisboa

Faculdade de Ciências da Universidade de Lisboa

Programação Paralela e Concorrente

José Eduardo Madeira
fc51720

Assignment 1

A maneira de como paralelizei o problema, foi dividindo o DNA recebido pelo numero máximo de threads disponíveis na maquina, e cada thread verificar se cada char daquele array de DNA esta presente em algum pattern dos quais estamos à procura, se estiver, voltar a verificar se os seguintes chars pertencem a esse mesmo pattern(caso o pattern tenha mais que um char), se se verificar o aparecimento de algum pattern, incrementamos o valor de aparecimento do mesmo , tendo sido criado um array results com o mesmo tamanho que a lista de patterns assim sempre que encontramos um pattern na posição x da sua lista , ira ser incrementado um valor na posição x do array.

Devido a esse array results, tive concurrency issues, pois todas as threads estavam a partilhar memória e a alterar este array ao mesmo tempo , daí ter usado o comando `synchronized(results) {...}` para evitar ter esses mesmos problemas.

8 cores machine :
(média) Tseq≈7467 SpeedUp=Tseq/Tpar= 0,01532378
(média) Tpar≈487282 Occupancy=SpeedUp/nCores= 0,00191547

64 cores machine :
(média) Tseq≈437928 SpeedUp=Tseq/Tpar= 0,01414014
(média) Tpar≈30970548 Occupancy=SpeedUp/nCores= 0,00022094

Analisando estes dados, concluo que a solução paralelizada é bastante mais lenta do que a solução sequencial, pois a criação de threads é muito mais custosa e lenta do que percorrer uma lista. Mas como visto, num exemplo parecido, na aula, haverá um certo tamanho de lista de dna que a solução paralelizada será melhor.