Faculdade Nova Roma
Técnicas Avançadas de Programação
Semestre: 2014.2
Professor: Allan Diego Silva Lima

Aluno: Tarcisio Deschamps Silva

Avaliação Individual

Valor Total: 12 pontos

Todas as respostas devem ser escritas na sua folha de papel pautado e serem feitas usando caneta.
Não se esqueça de colocar o seu nome nele também.

1) A distância Euclidiana entre dois pontos p e q é o tamanho do segmento de reta que conecta ambos. Em coordenadas cartesianas, sejam p = (p1, p2, ... , pn) e q = (q1, q2, ... , qn) dois pontos em um espaço euclidiano de n dimensões. A distância entre p e q é dada por:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

O conceito ilustrado acima tem diversas aplicações na computação moderna. Entre suas elas é possível destacar como exemplo problemas relacionados à Computação Gráfica e à Recuperação de Informação.

a) (2.0) Imagine que você foi alocado para paralelizar a computação da distância Euclidiana entre diversos vetores. O problema evolve a computação da distância Euclidiana entre 2000 (dois mil) vetores distintos (gerando 1000 resultados) com 10 000 (dez mil) dimensões cada. Elabore uma estratégia de paralelização para este problema. Considere que o computador no qual onde sua estratégia será utilizada seja capaz de executar **n threads** em paralelo.

b) (1.0) Há a possiblidade de ocorrer *deadlock* ou *starvation* na sua solução? Em caso afirmativo, como evitar tais problemas?

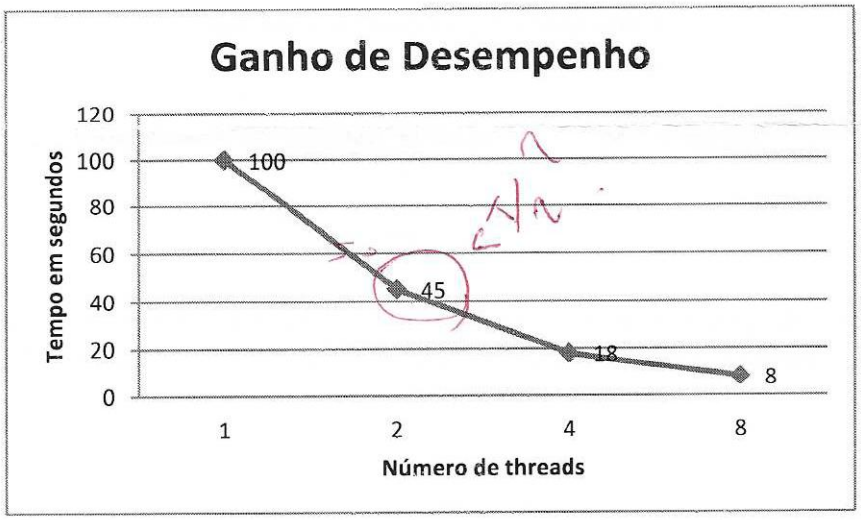
c) (1.0) Escreva o pseudocódigo da sua solução (priorize a distribuição do processamento entre threads, processamento dos dados por cada thread e eventuais seções críticas que possam existir).

2) (2.0) Defina os conceitos de *semáforo* e *mutex*, descreva também as principais diferenças entre ambos.

3) (2.0) Dê um exemplo de paralelismo e um exemplo de concorrência (em nível de instrução de código). Comente a relação entre ambos.

4) (2.0) Comente sobre as consequências teóricas em termos de **memória** e **desempenho** da utilização de um *buffer* durante operações de entrada de saída.

5) (2.0) João foi alocado para a tarefa de paralelização do principal gargalo no desempenho temporal de um dos projetos da sua empresa. Após uma semana de trabalho, ele entrega um relatório ao seu gerente contendo o seguinte gráfico:



Caso você fosse o gerente de João, como avaliaria o ganho obtido pela solução por ele criada?

FACULDADE NOVA ROMA

NOME: Tarcisio Deschamps Silva

Data _____

9,4

1) a) dividir linhas de (p, q) por n // n = numero de thread
thread (n) { recebe linha (p, q);
calculo d(p, q);
synchronized {
atribui resultado em vetor resposta } 0,2

b) Starvation ocorre em caso de o numero de threads ~~estiv~~ estiverem lotados e por algum erro de implementação não conseguirem terminar de executar impedindo que as outras linhas executem d(p, q)

FACULDADE NOVA ROMA

NOME: Tarcisio Deschamps Silva

6,4

Data

1,8

2:) • Semáforo: Identificador inteiro responsável por garantir acesso a região crítica

• Mutex: mantem acesso a região crítica sincronizado, podendo ter mais de uma Thread em execução mas só um tendo acesso a RC. O que no semáforo só deixa ele ser executado quando tiver acesso a RC

1,5

4:) Ao utilizar o buffer você tem um ganho de desempenho pois irá diminuir a quantidade de operações de E/S em disco, que são bem lentas, e carrega o maximo possível para o buffer em uma unica operação, só realizando a E/S novamente quando o buffer estiver vazio.

1,0

5:) Pois entre 4 à 8 nucleos já teria alcançado o máximo de vantagens do uso da paralelização, e que se aumentasse o numero de threads mais ainda (>8) não traria uma redução de tempo significativa que valesse o investimento em mais threads. E nenhum programa é capaz de rodar em zero segundos.

1,5

3:) • Paralelismo

```
int n=0
    thread1.soma(int i){
        n+=i;}
    thread2.soma(int i){
        n+=i;}
    print(n)
```

} nesse modo ele pega o valor de n em cada Thread e incrementa i o que torna não deterministico pois se as executarem os dois threads simultaneamente (paralelo) ambas pegarão n=0 e sairá n=0 caso de i=1, n=1 duas vezes

• Concorrência

```
int n=0
private int soma(int i){
synchronized; n+=i; }
    thred1.soma(2)
    thred2.soma(4)
    print(n)
```

} ha concorrência eles tentam acessar um recurso compartilhado que só pode ser acessado em uma Thread por vez. O synchronized serve para garantir que só uma obtenha acesso ao recurso
Resultando n=6

Relação entre ambos: P ⊂ C