2023.2 Programação Concorrente

Luigi Rangel, DRE: 121077746

27/08/2023

Questão 1. *No programa de multiplicação de matrizes mostramos (Cap 2 - texto) uma forma de paralelizar o algoritmo de multiplicação de matrizes criando um fluxo de execução independente para calcular cada um dos elementos da matriz de saída. Proponha outra solução onde a tarefa de cada fluxo de execução seja calcular uma linha inteira da matriz de saída.*

Resposta:

```c
#define N 1000 //N igual a dimensao da matriz

float a[N][N], b[N][N], c[N][N];

void calculaLinhaMatriz(int dim, int i) {
    int j, k, soma;
    for(j = 0; j < dim; j++) {
        soma = 0;

        for(k = 0; k < dim; k++) {
            soma += a[i][k] * b[k][j];
        }

        c[i][j] = soma;
    }
}
void main() {
    int i;
    //inicializa as matrizes a e b (...)
    //faz C = A * B
    for(i = 0; i < N; i++) {
        //dispara um fluxo de execucao f para executar:
        //calculaLinhaMatriz(N, i);
    }
}
```

Questão 2. *Para arquiteturas de hardware com poucas unidades de processamento (como é o caso das CPUs multiicores) geralmente é melhor criar uma quantidade de fluxos de execução igual ao número de unidades de processamento. Altere a solução do exercício anterior fixando o numero de fluxos de execução e dividindo o cálculo das linhas da matriz de saída entre eles.*

Resposta:

```c
#define N 1000 //N igual a dimensao da matriz
#define P 4 //P igual ao numero de unidades de processamento

float a[N][N], b[N][N], c[N][N];

void calculaSegmentoMatriz(int dim, int ini, int fim) {
    int i, j, k, soma;
    for (i = ini; i < fim; i++) {
        for(j = 0; j < dim; j++) {
            soma = 0;

            for(k = 0; k < dim; k++) {
                soma += a[i][k] * b[k][j];
            }

            c[i][j] = soma;
        }
    }
}
void main() {
    int i, inicio, fim;
    //inicializa as matrizes a e b (...)
    //faz C = A * B
    for(i = 0; i < P; i++) {
        inicio = (N * i) / P;
        fim = N * (i + 1) / P;
        //dispara um fluxo de execucao f para executar:
        //calculaSegmentoMatriz(N, inicio, fim);
    }
}
```

Questão 3. *A série mostrada abaixo pode ser usada para estimar o valor da constante π. A função* **`piSequencial()`** *implementa o calculo dessa série de forma sequencial. Proponha um algoritmo concorrente para resolver esse problema dividindo a tarefa de estimar o valor de π entre M fluxos de execução independentes.*

$$\pi = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots)$$

```c
double piSequencial (long long n) {
    double soma = 0.0, fator = 1.0;
    long long i;
    for (i = 0; i < n; i++) {
        soma = soma + fator/(2*i+1);
        fator = -fator;
    }
    return 4.0 * soma;
}

```

Resposta:

```c
#define N 100 //N igual a quantidade de termos da serie a serem calculados
#define P 4 //P igual a quantidade de unidades de processamento

void calculaSegmentoPi(double *seg, int ini, int fim) {
    int i, fator = 1 - 2 * (ini % 2);
    double soma = 0;

    for (i = ini; i < fim; i++) {
        soma += fator / (2 * i + 1);
        fator *= -1;
    }

    *seg = soma;
}

void main() {
    int i, inicio, fim;
    double pi = 0;
    double *segs;
    //aloca P espacos de double na memoria para a variavel segmentos

    for (i = 0; i < P; i++) {
        inicio = N * i / P;
        fim = N * (i + 1) / P;
        //dispara um fluxo de execucao f para executar
        //calculaSegmentoPi(&segs[i], inicio, fim);
    }

    //espera o fim de todos os fluxos de execucao

    for (i = 0; i < P; i++) {
        pi += segs[i];
    }
    pi *= 4;
}
```

Questão 4. *A série infinita mostrada abaixo estima o valor de* $log(1+x)$ $(-1 < x <= 1)$*:*

$$\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \dots$$

Dois programas foram implementados para calcular o valor dessa série (um programa sequencial e outro concorrente) usando N termos. Apos a implementação, foram realizadas execuções dos dois programas, obtendo as medidas de tempo apresentadas na Tabela 1. A coluna N informa o numero de elementos da série, a coluna thread informa o número de threads, e as colunas T_s *e* T_c *informam os tempos de execução do programa sequencial e do programa concorrrente, respectivamente.*

N	threads	Ts (s)	Tc (s)	A
1×10^6	1	0,88	0,89	
1×10^6	2	0,88	0,50	
1×10^7	1	8,11	8,34	
1×10^7	2	8,11	4,44	
2×10^7	1	16,21	16,41	
2×10^7	2	16,21	8,84	

Table 1: Medidas de tempo para calcular $\log(1+x)$.

a. *Complete a coluna A com os valores de aceleração.*

Resposta:

N	threads	Ts (s)	Tc (s)	A
1×10^6	1	0,88	0,89	0,99
1×10^6	2	0,88	0,50	1,76
1×10^7	1	8,11	8,34	0,97
1×10^7	2	8,11	4,44	1,82
2×10^7	1	16,21	16,41	0,99
2×10^7	2	16,21	8,84	1,83

Table 2: Aceleração preenchida.

b. *Avalie os resultados obtidos para essa métrica. Considere os casos em que a carga de dados aumenta junto com o número de processadores e os casos isolados onde apenas a carga de trabalho ou o número de processadores aumenta.*

Resposta:

Quando o número de processadores aumenta, mas a carga de trabalho não, há aceleração sublinear. Entretanto, quando a carga de trabalho aumenta mas o número de processadores não, a aceleração permanece quase igual. Ou seja, o programa paralelo se mostra útil neste caso para ser implementado em arquiteturas com 2 ou mais processadores.

Questão 5. *Considere uma aplicação na qual* 20% *do tempo total de execução é comprometido com tarefas sequenciais e o restante,* 80%, *pode ser executado de forma concorrente.*

a. *Se dispusermos de uma máquina com 4 processadores, qual será a aceleração teórica (de acordo com a lei de Amdahl) que poderá ser alcançada em uma versão concorrente da aplicação?*

Resposta:

A aceleração teórica será de $\frac{1}{0.2+(0.8/4)} = 2.5$.

b. *Se apenas* 50% *das atividades pudessem ser executadas em paralelo, qual seria a aceleração teórica considerando novamente uma máquina com 4 processadores?*

Resposta:

A aceleração teórica seria de $\frac{1}{0.5+(0.5/4)} = 1.6$.
