Computação Concorrente (DCC/UFRJ)

Aula 4: Comunicação entre threads via memória compartilhada (parte 1)

Prof. Silvana Rossetto

3 de setembro de 2019

Exercício (15 min)

Descrição do problema

Dado um vetor de números reais (float) com N elementos, implemente um programa concorrente para somar todos os elementos desse vetor e exibir o resultado na saída padrão.

Requisitos do problema

- o número de threads deve ser recebido como parâmetro de entrada
- a solução deve garantir balanceamento de carga entre as threads

Estratégias para divisão das tarefas entre as threads

- as threads somam elementos intercalados (Solução 1)
- as threads somam blocos consecutivos de elementos (Solução 2)
- as threads somam blocos consecutivos intercalados (Solução 3)
- ...

Variáveis globais

```
//tamanho do vetor
int tam;
//vetor de elementos
float *vet;
//vetor com os resultados das somas de cada thread
float *soma;
//numero de threads
int nthreads;
```

Solução 1: código das threads

```
void *SomaVetor1 (void *tid) {
  int id = * (int *) tid;
  float soma_local = 0; //guarda soma local
  float *ret;  //retorno com pthread_exit
  ret = malloc(sizeof(float));

  //soma os valores
  for(int i=id; i<tam; i+=nthreads) {
     soma_local += vet[i];
  }

  //retorna o resultado da soma
  soma[id] = soma_local; //opcao 1: retorno global
  *ret = soma_local; //opcao 2: retorno com pthread_exit
  pthread_exit((void*) ret);
}
```

Solução 1: código da main

```
float *retorno;
...
for(t=0; t<nthreads; t++) {
   //pthread_join(tid_sistema[t], NULL);
   pthread_join(tid_sistema[t], (void *) &retorno);

   soma_total_op1 += soma[t];  //opção 1

   soma_total_op2 += *retorno; //opçao 2

   free(retorno);
}
```

Ver execução do código (solução 1)...

- O resultado da soma sequencial (do primeiro até o último elemento) pode ser diferente do resultado da soma concorrente? Por que?
- Em qual situação os resultados serão os mesmos?

Solução 2: código das threads

```
void *SomaVetor2 (void *tid) {
  int id = * (int *) tid;
  float soma_local = 0;   int ini, fim, bloco;

  bloco = tam/nthreads; //tamanho de cada bloco
  ini = id*bloco; //posicao inicial do vetor
  fim = ini + bloco; //posicao final do vetor
  //a ultima thread trata os elementos restantes
  if (id==(nthreads-1)) fim = tam;

  //soma os valores
  for(int i=ini; i<fim; i++) {
     soma_local += vet[i];
  }
  soma[id] = soma_local; //opcao 1: retorno global
}
```

Ver execução do código (solução 2)...

- O resultado da soma sequencial (do primeiro até o último elemento) pode ser diferente do resultado da soma concorrente nessa solução? Ou nesse caso a possibilidade de erro é menor?
- Em qual situação os resultados serão os mesmos?

Solução incorreta!!!

```
//soma total (global)
float soma=0;

void *SomaVetor (void *tid) {
  int id = * (int *) tid, i; free(tid);
  //float soma_local = 0;

  //soma os valores
  for(i=id; i<tam; i+=nthreads) {
     //soma_local += vet[i];
     soma += vet[i];
  }

  //soma += soma_local;
  pthread_exit(NULL);
}
```

Ver execução do código incorreto!!!

- O que ocorre com o resultado calculado pelas threads?
- Por que essa solução está incorreta?

Exercício

Considere a definição do valor de π mostrada abaixo. Como você implementaria um programa concorrente para realizar esse cálculo?

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + (-1)^n \frac{1}{2n+1} + \cdots\right)$$