Lista de Revisão II

Programação Concorrente e Distribuída

16 de outubro de 2015

1. Em CUDA, suponha uma aplicação em que cada thread deve calcular um elemento de saída de uma adição de vetores. Qual seria a expressão para calcular o índice de acesso aos elementos do vetor de saída com base nos índices das threads e blocos? Apresente uma breve explicação sobre a fórmula em relação à Grid de threads?

   (a) `i=threadIdx.x + threadIdx.y;`

   (b) `i=blockIdx.x + threadIdx.x;`

   (c) `i=blockIdx.x*blockDim.x + threadIdx.x;`

   (d) `i=blockIdx.x * threadIdx.x;`

2. Para uma adição de vetores, suponha o tamanho dos vetores seja de 8000 elementos cada, que cada thread calcula um elemento de saída, e que o tamanho do bloco é de 1024 threads. Qual o número mínimo de threads necessárias para cobrir todas as posições do vetor? Por que?

   (a) 8000

   (b) 8196

   (c) 8192

   (d) 8200

3. Para uma adição de vetores, suponha que o tamanho dos vetores seja de 1023 elementos cada. Considerando que cada thread calcula um elemento de saída e que o tamanho do bloco é de 128 threads. Qual o número mínimo de threads necessárias para cobrir todas as posições do vetor? **Marque uma alternativa e explique a sua resposta.**

   (a) 1023

   (b) 1024

   (c) 2048

   (d) 128

4. Identifique em qual código o padrão SCAN pode ser aplicado:

   (a) `S[i] = S[i-1] * A[i]`

   (b) `soma = soma + A[i]`

   (c) `S[i] = A[i] + B[i]`

   (d) `S[i] = A[i-1] + B[i-1]`

5. Indique qual o padrão paralelo (MAP, REDUCE ou SCAN) que se encaixa em cada um dos códigos abaixo, onde i e k são variáveis de controle de loops:

(a) `S[i] = S[i-1] * A[i]` Padrão:

(b) `soma = soma + A[i]` Padrão:

(c) `S[i] = A[i] + B[i]` Padrão:

(d) `S[i] = A[i-1] + B[i-1]` Padrão:

(e) `S[i] *= A[k] * B[k]` Padrão:

(f) `S[i] = S[i-1]` Padrão:

6. Implemente um kernel em CUDA, onde dado um vetor de entrada `input`, com `n` elementos, é gerado um vetor de saida `output`, onde cada posição `i` da saída é o quadrado de `input[i]`.

7. A chamada de `_syncthreads()` é uma barreira que serve para:

   (a) Sincronizar todas as threads de um kernel.

   (b) Sincronizar threads que compartilham um mesmo dado.

   (c) Sincronizar todas as threads dentro de uma mesmo bloco.

   (d) Sincronizar threads que acessam dados diferentes.

8. Suponha que um kernel seja lançado com 1000 blocos de threads, sendo que cada bloco tenha 512 threads. Se uma variável compartilhada (`__shared__`) é declarada, quantas instâncias desta variável serão criadas durante a execução do kernel? **Marque uma alternativa e explique a sua resposta.**

   (a) 1

   (b) 1000

   (c) 512

   (d) 512000