Iniciado em	segunda-feira, 11 dez. 2023, 14:10
Estado	Finalizada
Concluída em	segunda-feira, 11 dez. 2023, 15:22
Tempo empregado	1 hora 11 minutos
Avaliar	**6,00** de um máximo de 10,00(**60**%)

Questão 1

Completo

Atingiu 2,00 de 2,00

Elabore um programa MPI com N processos, sendo o master o responsável por inicializar o vetor e os slaves, responsáveis por imprimir uma porção do vetor, proporcional ao número de slaves identificados pelo programa.

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

#define VEC_SIZE 20

int main(int argc, char *argv[]) {
  int rank, size;
  int *vector = NULL;
  int portion_size, remainder, local_size;

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  if (rank == 0) {
    // O processo mestre inicializa o vetor
    vector = (int *)malloc(VEC_SIZE * sizeof(int));
    for (int i = 0; i < VEC_SIZE; i++) {
      vector[i] = i + 1;
    }

    // Envia porções do vetor para cada processo escravo
    for (int dest = 1; dest < size; dest++) {
      portion_size = VEC_SIZE / (size - 1);
      remainder = VEC_SIZE % (size - 1);
      local_size = portion_size + (dest <= remainder ? 1 : 0);

      MPI_Send(&local_size, 1, MPI_INT, dest, 0, MPI_COMM_WORLD);
      MPI_Send(&vector[(dest - 1) * portion_size + (dest - 1 < remainder ? dest - 1 : remainder)],
          local_size, MPI_INT, dest, 0, MPI_COMM_WORLD);
    }
    printf("Slave (rank %d) iniciou o vetor\n", rank);
    free(vector);
  } else {
    // Os processos escravos recebem suas porções do vetor
    MPI_Recv(&local_size, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    int *local_vector = (int *)malloc(local_size * sizeof(int));
    MPI_Recv(local_vector, local_size, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

    // Os processos escravos imprimem suas porções do vetor
    printf("Slave (rank %d) faz parte do vetor: ", rank);
    for (int i = 0; i < local_size; i++) {
      printf("%d ", local_vector[i]);
    }
    printf("\n");

    free(local_vector);
  }

  MPI_Finalize();
  return 0;
```

}

Comentário:

Questão 2

Correto

Atingiu 1,00 de 1,00

O código a seguir foi compilado com OpenMP e o binário tem o nome t1. Com base no código fonte, analise as afirmações e marque V para as verdadeiras e F para as falsas.

```
1   #include <stdio.h>
2   #include <omp.h>
3
4   int main(int argc, char *argv[]) {
5
6           int i=0;
7           #pragma omp parallel
8           {
9                   if (omp_get_thread_num() == 1)
10                          i=i+10;
11          }
12          printf("i=%d\n", i);
13          return 0;
14  }
```

I - A execução com o comando **OMP_NUM_THREADS=1 t1** vai imprimir o valor zero para a variável i

II - Se na linha 7 for acrescentada a declaração **private(i)**, o binário equivalente acionado com o comando **OMP_NUM_THREADS= 2 t1** vai imprimir o valor 20

III - Ao suprimir a linha 9, o binário equivalente vai imprimir valores aleatórios para a variável i, desde que o número de threads seja maior que 1

a. Apenas as afirmativas I e III estão corretas ✔

b. Nenhuma das opções apresentadas consegue julgar corretamente as afirmativas

c. Apenas a afirmativa I está correta

d. Apenas as afirmativas I e II estão corretas

e. Apenas as afirmativas II e III estão corretas

Sua resposta está correta.

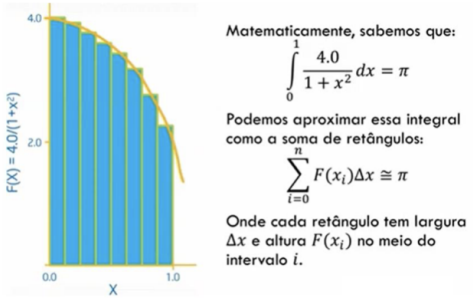
A resposta correta é:
Apenas as afirmativas I e III estão corretas

Questão 3

Completo

Atingiu 2,00 de 2,00

O programa a seguir é uma implementação do cálculo da função Pi em modo serial, usando único espaço de endereçamento. Com base neste código crie uma versão CUDA, de modo que o valor de PI seja obtido por cálculos realizados na GPU, obedecendo a melhor relação possível entre blocos e threads por bloco.



Matematicamente, sabemos que:

$$\int_0^1 \frac{4.0}{1+x^2}\,dx = \pi$$

Podemos aproximar essa integral como a soma de retângulos:

$$\sum_{i=0}^{n} F(x_i)\Delta x \cong \pi$$

Onde cada retângulo tem largura Δx e altura $F(x_i)$ no meio do intervalo i.

```c
#include <stdio.h>
#define NUM_STEPS 8000000

int main(void) {
    double x, pi, sum=0.0;
    double step;

    step = 1.0/(double) NUM_STEPS;
    for (int i=0;i<NUM_STEPS; i++) {
        x = (i+0.5) * step;
        sum+=4/(1.0+x*x);
    } /*fim-for */
    pi = sum*step;
    printf("Pi = %f\n", pi);
} /*fim-main */
```

```c
#include <stdio.h>
#include <cuda_runtime.h>

#define NUM_STEPS 8000000
#define BLOCK_SIZE 256

__global__ void calculatePi(double *sum_dev) {
  int i = blockIdx.x * blockDim.x + threadIdx.x;
  double x;
  double step = 1.0 / (double)NUM_STEPS;

  if (i < NUM_STEPS) {
    x = (i + 0.5) * step;
    sum_dev[i] = 4.0 / (1.0 + x * x);
  }
}

int main() {
  double *sum_dev;
  double pi;
  double step = 1.0 / (double)NUM_STEPS;

  cudaError_t cudaStatus;

  // Aloca memoria para a soma
  cudaStatus = cudaMalloc((void **)&sum_dev, NUM_STEPS * sizeof(double));
  if (cudaStatus != cudaSuccess) {
    fprintf(stderr, "cudaMalloc failed: %s\n", cudaGetErrorString(cudaStatus));
    return 1;
```

```
    }

    // Inicia o kernel para calcular a soma
    int num_blocks = (NUM_STEPS + BLOCK_SIZE - 1) / BLOCK_SIZE;
    calculatePi<<<num_blocks, BLOCK_SIZE>>>(sum_dev);

    cudaStatus = cudaGetLastError();
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "calculatePi launch failed: %s\n", cudaGetErrorString(cudaStatus));
        return 1;
    }

    // Copia os resultados de volta para o host
    double *sum = (double *)malloc(NUM_STEPS * sizeof(double));
    cudaStatus = cudaMemcpy(sum, sum_dev, NUM_STEPS * sizeof(double), cudaMemcpyDeviceToHost);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaMemcpy failed: %s\n", cudaGetErrorString(cudaStatus));
        return 1;
    }

    // Calcula o final da soma no host
    pi = 0.0;
    for (int i = 0; i < NUM_STEPS; i++) {
        pi += sum[i];
    }
    pi *= step;

    printf("Pi = %f\n", pi);

    free(sum);
    cudaFree(sum_dev);

    return 0;
}
```

q3.cu

Comentário:

Questão 4

Completo

Atingiu 0,00 de 2,00

O programa abaixo trabalha com vetores de tamanho variável localmente. Com uso da biblioteca RPC, gere uma estrutura equivalente na qual o programa principal e a função chamada (somavet) residam em hosts distintos.

Obs.: Sugere-se criar uma pasta contendo os arquivos da sua aplicação. Essa pasta deve ser compactada e enviada como resposta para essa questão.

```
1  int somavet(int *x, int n) {
2        int i, result;
3        printf("Requisicao para adicao de %d numeros\n", n);
4        result=0;
5        for (i=0; i<n; i++) {
6          result += x[i];
7        } /* fim-for */
8        return (result);
9  } /* fim somavet */
10 int main( int argc, char *argv[]) {
11   int *ints, n_termos, i, res;
12   if (argc<2) {
13     fprintf(stderr,"Uso correto: %s num1 num2 ...\n",argv[0]);
14     exit(0); }
15     /* recupera os numeros que devem ser adicionados */
16   n_termos = argc-1;
17   ints = (int *) malloc(n_termos * sizeof( int ));
18   if (ints==NULL) {
19     fprintf(stderr,"Erro na alocacao de memoria\n");
20     exit(0); }
21 /* preenche a estrutura dinamica com os valores informados pelo usuario */
22   for (i=1;i<argc;i++) {
23     ints[i-1] = atoi(argv[i]);
24   } /* fim-for */
25 /* imprime o resultado da soma */
26   res = somavet(ints, n_termos);
27   printf("%d",ints[0]);
28   for (i=1; i<n_termos; i++) {
29     printf(" + %d",ints[i]);
30   } /* fim-for */
31   printf(" = %d\n",res);
32   return(0);
33 } /* fim programa principal */
34
35
36
```

A resposta segue compactada

q4_lucas_gomes_212005426.zip

Comentário:

Questão 5

Correto

Atingiu 1,00 de 1,00

O código a seguir foi compilado com OpenMP e o binário tem o nome t1. Com base no código fonte, analise as afirmações e marque V para as verdadeiras e F para as falsas.

```
1   #include <stdio.h>
2   #include <omp.h>
3
4   int main(void) {
5           printf(" %d\n", omp_get_num_threads());
6           #pragma omp parallel
7           {
8
9           }
10          printf("%d\n", omp_get_max_threads());
11          return 0;
12  }
```

I - A execução com o comando **OMP_NUM_THREADS=4 t1** vai imprimir o valor 4 na linha 5 e o valor 12 na linha 10, se o computador onde esse programa estiver rodando tiver 12 núcleos

II - Este programa vai imprimir sempre o valor 1 na linha 5, independente do número de threads definidas na variável OMP_NUM_THREADS

III - O comando da linha 10 vai imprimir sempre o valor 1, uma vez que este está fora da região paralela definida pelo **pragma omp parallel**

a. Apenas as afirmativas I e II estão corretas

b. Apenas as afirmativas I e III estão corretas

c. Apenas as afirmativas II e III estão corretas

d. Nenhuma das opções consegue julgar as afirmativas apresentadas ✔

e. Apenas a afirmativa I está correta

Sua resposta está correta.

A resposta correta é:
Nenhuma das opções consegue julgar as afirmativas apresentadas

Questão 6

Incorreto

Atingiu 0,00 de 1,00

Analise as afirmativas abaixo e marque a opção correta

I - No modelo P2P, o DHT (***Distributed Hash Table***) é uma técnica de localização de peers que se caracteriza por ter boa performance em redes com grande quantidade de peers que podem sair e entrar na rede a qualquer instante

II - O modelo P2P é um tipo de arquitetura na qual os peers são máquinas que possuem a função de serem servidoras e clientes ao mesmo tempo, como uma se fosse uma variante do modelo Cliente/Servidor tradicional das aplicações de rede TCP/IP

III - Algoritmos de consenso derivados do Paxos, como o Zookeeper e o Raft tem, como uma de suas funções, a sincronização dos logs de execução entre os peers de uma rede P2P

a. Apenas I e II estão corretas ✖

b. Apenas III está correta

c. Apenas II está correta

d. Apenas II e III estão corretas

e. Apenas I está correta

Sua resposta está incorreta.

A resposta correta é:
Apenas III está correta

Questão 7

Incorreto

Atingiu 0,00 de 1,00

Analise o código a seguir.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <cuda_runtime.h>
4  #include <cuda.h>
5
6  __global__ void vecAdd(int *c) {
7      int i=threadIdx.x;
8      int j=blockIdx.x;
9      int k=blockDim.x;
10
11     c[i] = i+j+k;
12 } /* fim vecAdd */
13
14 int main(int argc, char *argv[]) {
15     int *c, *dc;
16     int n=atoi(argv[1]);
17     int size = n * sizeof(int);
18
19     cudaDeviceReset();
20     c = (int*) malloc(size);
21     cudaMalloc((void **) &dc, size);
22     vecAdd <<<2,n/2>>> (dc);
23     cudaDeviceSynchronize();
24     cudaMemcpy(c, dc, size, cudaMemcpyDeviceToHost);
25
26     printf("Resultado:\n");
27     for (int i=0; i<n; i++)
28         printf(" %d ", c[i]);
29
30     printf("\n");
31     cudaFree(dc);
32
33     return 0;
34 } /* fim-main */
```

Suponha que n (linha 16) é igual a 10, analise as afirmativas a seguir e marque a alternativa INCORRETA.

a. Da forma como está, o vetor a ser impresso é 6 7 8 9 10 0 0 0 0 0 ✖

b. Se houver substituição do comando da linha 11 por c[j] = i+j+k;, o vetor a ser impresso é 5 6 0 0 0 0 0 0 0 0

c. Se for utilizada a fórmula c[i+(j*k)]=i+j+k, o vetor a ser impresso é 0 0 0 0 0 5 6 7 8 9

d. Se houver substituição do comando da linha 11 por c[k] = i+j+k;, o vetor a ser impresso é 0 0 0 0 0 6 0 0 0 0

e. Nenhuma das alternativas está ERRADA

Sua resposta está incorreta.

A resposta correta é:
Se for utilizada a fórmula c[i+(j*k)]=i+j+k, o vetor a ser impresso é 0 0 0 0 0 5 6 7 8 9