Iniciado em	segunda-feira, 16 set. 2024, 14:34
Estado	Finalizada
Concluída em	segunda-feira, 16 set. 2024, 15:23
Tempo empregado	48 minutos 57 segundos
Avaliar	**5,00** de um máximo de 10,00(**50**%)

Questão **1**

Completo

Atingiu 0,00 de 2,00

Elaborar um microserviço MPI (linguagem C), no qual o master (que oferece o serviço) receba requisições dos slaves e contabilize palavras em um dicionário da seguinte forma:

- se palavra_recebida = IMPRIMIR

  listar o conteúdo do dicionário (cada palavra e o número de ocorrências)
- senão

  se palavra_recebida existe no dicionário:
  incrementar o contador de palavras relativo à palavra_recebida
  senão
  incluir a palavra_recebida no dicionário
  contador de palavras de palavra_recebida = 1

Por sua vez, a função consumidora do microserviço (função main) deve ter os seguintes modos:

- Modo inclusão:

  obter as palavras a serem contabilizadas a partir de um arquivo de entrada
  enviar cada palavra identificada para o microserviço remoto
- Modo consulta:

  enviar a string IMPRIMIR para o microserviço remoto

Na resposta, entregar arquivo compactado contendo: os códigos .c do cliente e do servidor da aplicação, e um README com identificação do aluno (matrícula/nome) e instruções de execução

Envio do arquivo junto com README na atividade aberta

Comentário:
Estrutura errada! Não funciona

Questão 2

Completo

Atingiu 0,00 de 2,00

No código a seguir, os pragmas declarados nas linhas 11 e 14 garantem a divisão equilibrada do trabalho entre o total de threads especificadas na variável de ambiente OMP_NUM_THREADS.

```
1  #include <stdio.h>
2  #include <omp.h>
3  #define TAM 12
4  int main () {
5      int A[TAM], B[TAM], C[TAM];
6      int i;
7      for (i=0; i<TAM; i++) {
8          A[i]=2*i - 1;
9          B[i]=  i + 2;
10     }
11     #pragma omp parallel
12     {
13         int tid = omp_get_thread_num();
14         #pragma omp for
15         for (i=0; i<TAM; i++) {
16             C[i] = A[i]+ B[i];
17             printf("Thread[%d] calculou C[%d]\n", tid, i);
18         } /* fim-for */
19     } /* fim-pragma */
20     for (i=0; i<TAM; i++)
21         printf("C[%d]=%d\n", i, C[i]);
22 } /* fim-main */
23
```

Apresente uma nova versão desse código que garanta a distribuição equilibrada de trabalho entre as threads (de acordo com o valor de OMP_NUM_THREADS), considerando apenas o pragma de paralelização descrito na linha 11 (ou seja, assuma a não existência do pragma da linha 14).

Readme e codigo no zip

p3_OMP_DaniloDomingo_180015311.zip

Comentário:
Não funciona. Vc não isolou o índice do laço

Questão 3

Correto

Atingiu 1,00 de 1,00

Julgue as afirmações abaixo e marque a alternativa correta:

I - No Hadoop, o número de instâncias de funções map() é equivalente ao número de chunks que o HDFS promoveu no(s) arquivo(s) de entrada

II - No Spark, a função **fold (1, lambda x, y: x+y)** aplicada a um RDD contendo a lista [1, 2, 3, 4, 5] produzirá o resultado 24 se o número de partições do referido RDD for igual a 8

III - No Hadoop, o número de arquivos produzidos na pasta de saída é sempre igual ao número de funções reduce() instanciados na aplicação

a. Apenas a afirmativa I está correta

b. Apenas a afirmativa II está correta

c. Apenas a afirmativa III está correta

d. Nenhuma das opções satisfaz as afirmativas apresentadas ✔

e. Apenas as afirmativas I e II estão corretas

Sua resposta está correta.

I - correto

II - correto

III - correto

A resposta correta é:
Nenhuma das opções satisfaz as afirmativas apresentadas

Questão 4

Correto

Atingiu 1,00 de 1,00

Analise as afirmativas a seguir e marque a alternativa correta.

I - Em programas concebidos de acordo com o paradigma Map/Reduce, cabe ao programador a tarefa de distribuir os serviços entre os nós do cluster

II - No paradigma Map/Reduce os dados a serem processados são enviados onde os códigos Map e Reduce estão instalados, a fim de promover a melhora de desempenho e o paralelismo desejado.

III - Uma das desvantagens das infra-estruturas que fazem uso do Map/Reduce com HDFS é o grande consumo de tempo com operações de I/O em discos (memória secundária).

a. Apenas I e III estão corretas

b. Nenhuma das respostas

c. Apenas II e III estão corretas

d. Apenas I está correta

e. Apenas III está correta ✔

Sua resposta está correta.

A resposta correta é:
Apenas III está correta

Questão 5

Correto

Atingiu 1,00 de 1,00

Julgue as afirmações abaixo:
I - Sockets UDP, por serem não orientados à conexão, permitem a comunicação persistente entre processos cliente e servidor
II - Sincronicidade é uma das funcionalidades atendidas pela biblioteca MPI, uma vez que esta garante a entrega da mensagem no receptor, mesmo que o processo destinatário não esteja executando
III - Brokers como Kafka e RabbitMQ são interessantes para viabilizar comunicação persistente entre processos

a. Apenas a afirmação III está correta ✔

b. Apenas as afirmações I e III estão corretas

c. Apenas as afirmações II e III estão corretas

d. Apenas a afirmação I está correta

e. Todas as afirmações estão corretas

Sua resposta está correta.

A resposta correta é:
Apenas a afirmação III está correta

Questão 6

Incorreto

Atingiu 0,00 de 1,00

Observe o seguinte código MPI, cujo objetivo é conseguir gravar 100 elementos do vetor data em arquivo:

```
1 #include <mpi.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #define FILE_NAME "file.bin"
5 #define MAX 100
6
7 int main(int argc, char** argv) {
8   int rank, size;
9   MPI_Init(NULL, NULL);
10  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
11  MPI_Comm_size(MPI_COMM_WORLD, &size);
12  int data[MAX];
13  MPI_File fh;
14
15  int chunk = MAX / size;
16  int start = rank * chunk * sizeof(int);
17  if (rank == (size-1)) chunk+=(MAX%size);
18  for (int i = 0; i < chunk; i++)
19    data[i] = rank * chunk + i + 1;
20  MPI_File_open(MPI_COMM_WORLD, FILE_NAME, MPI_MODE_CREATE | MPI_MODE_WRONLY, MPI_INFO_NULL, &fh);
21  MPI_File_write_at(fh, start, data, chunk, MPI_INT, MPI_STATUS_IGNORE);
22  MPI_File_close(&fh);
23  MPI_Finalize();
24 } /* fim-main */
```

Considerando o propósito definido para o código, avalie as afirmativas e, a seguir, marque a opção correta:

I - O código não funciona adequadamente porque a função da linha 21 necessita um laço para garantir que cada processo faça a escrita dos elementos sob sua responsabilidade na posição correta do arquivo

II - Este código funciona adequadamente e a instrução da linha 17 garante que os valores sequenciais, de 1 a 100, no vetor data, independente do número de processos

III - O código apresentado não funciona adequadamente porque a função de escrita (linha 21) exige que o vetor a ser gravado seja dividido em partes iguais entre os processos MPI

a. Apenas as afirmativas I e III estão corretas

b. Apenas a afirmativa I está correta

c. Apenas a afirmativa III está correta

d. Nenhuma das alternativas está correta

e. Apenas a afirmativa II está correta ✖

Sua resposta está incorreta.

O código funciona bem da forma como está (afirmativa I é falsa). A afirmativa II é falsa porque não há gravação de todos os valores de 1 a 100 (no entanto, o comando equilibra a gravação entre o número de processos). Afirmativa III é falsa (a linha 17 garante que eventuais sobras do vetor sejam gravadas pelo último processo com a função MPI_File_write_at)

A resposta correta é:
Nenhuma das alternativas está correta

Questão 7

Correto

Atingiu 1,00 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. 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

b. Nenhuma das alternativas está ERRADA

c. 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

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

e. 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 ✔

Sua resposta está correta.

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

Questão 8

Correto

Atingiu 1,00 de 1,00

Analise as afirmativas a seguir e marque a alternativa correta.

I - A comunicação do tipo publish/subscribe é apropriada para distribuir serviços entre vários processos, mas possui restrições para implementar balanceamento de carga em clusters de servidores.

II - Brokers AMQP são flexíveis a ponto de permitirem que processos se comuniquem usando uma mesma fila/stream ou filas/streams separadas, dependendo do tipo do problema.

III - Um sistema publish/subscribe viabiliza buffers temporários e formatação de mensageria, que são mecanismos muito apropriados para comunicações do tipo transiente.

a. Apenas II e III estão corretas ✔

b. Apenas I está correta

c. Apenas III está correta

d. Apenas I e III estão corretas

e. Nenhuma das respostas

Sua resposta está correta.

A resposta correta é:
Apenas II e III estão corretas