Iniciado em	segunda-feira, 30 out. 2023, 14:10
Estado	Finalizada
Concluída em	segunda-feira, 30 out. 2023, 15:28
Tempo empregado	1 hora 17 minutos
Avaliar	**4,10** de um máximo de 10,00(**41**%)

Questão 1

Completo

Atingiu 1,50 de 3,00

Utilizando a biblioteca MPI, elabore um programa para somar os elementos de duas matrizes A e B, quadradas (int), para gerar uma matriz C, de acordo com as seguintes regras:

- O programa deve conter um processo master e dois workers, que deverão trabalhar em conjunto para garantir a realização de soma dos elementos das matrizes A e B
- Supor que as matrizes sejam de 16 posições (int) e as matrizes A e B devem ser inicializadas com valores randômicos
- As operações de soma devem ser distribuídas entre os workers, de modo que a primeira e a terceira linha das matrizes sejam processadas pelo primeiro worker e a segunda e a quarta linha sejam processadas pelo segundo worker
- A comunicação entre os processos master e workers deve ser feita utilizando especificamente as funções MPI_Send e MPI_Recv
- Ao final, a matriz C resultante deve ser impressa (em colunas, formato de matriz) pelo processo master

```
/*
Nome: Douglas da Silva Monteles - 190012200
Como compilar e executar:
$ mpicc questao1.c -o saida

$ mpirun -n 3 ./saida
*/

#include <stdio.h>
#include <mpi.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>

#define DEBUG 1
#define TAMANHO_MATRIX 4
#define MASTER 0

// define o tipo da mensagem
#define TAG 0

void iniciar_matrix(int matrix[TAMANHO_MATRIX][TAMANHO_MATRIX]) {
for (int i = 0; i < TAMANHO_MATRIX; i++) {
for (int j = 0; j < TAMANHO_MATRIX; j++) {
// Números aleatórios entre 0 e 99
matrix[i][j] = rand() % 100;
}
}
}

void imprimir_matrix(int matrix[TAMANHO_MATRIX][TAMANHO_MATRIX]) {
for (int i = 0; i < TAMANHO_MATRIX; i++) {
for (int j = 0; j < TAMANHO_MATRIX; j++) {
printf("%d\t", matrix[i][j]);
}
printf("\n");
}
}

void imprime_linha(int linha[TAMANHO_MATRIX]) {
for (int i = 0; i < TAMANHO_MATRIX; i++) {
printf("%d ", linha[i]);
}
}

int main(int argc, char **argv) {
```

```
MPI_Init(&argc, &argv);
// obtém o id do processo
int myRank;
int size;

// inicializa o rank
MPI_Comm_rank(MPI_COMM_WORLD, &myRank);

// define o tamanho do rank
MPI_Comm_size(MPI_COMM_WORLD, &size);

if (size != 3) {
printf("O programa deve conter 1 processo master e 2 processos workers. n = 3\n");
printf("mpicc questao1.c -o saida\nmpirun -n 3 ./saida\n");
exit(0);
}

int matrixA[TAMANHO_MATRIX][TAMANHO_MATRIX];
int matrixB[TAMANHO_MATRIX][TAMANHO_MATRIX];
int matrixC[TAMANHO_MATRIX][TAMANHO_MATRIX];

if (myRank == 0) {
// Estamos no MASTER
srand(time(NULL));

// Inicializa as matrizes A e B
iniciar_matrix(matrixA);
iniciar_matrix(matrixB);

printf("Matriz A:\n");
imprimir_matrix(matrixA);
printf("\n");

printf("Matriz B:\n");
imprimir_matrix(matrixB);
printf("\n");

int linha = 0;
// envio do vetor para cada worker
for (int worker_id = 1; worker_id < size; worker_id++) {
for (int i = 0; i < (TAMANHO_MATRIX / (size - 1)); i++) {
MPI_Send(matrixA[linha], TAMANHO_MATRIX, MPI_INT, worker_id, TAG, MPI_COMM_WORLD);
MPI_Send(matrixB[linha], TAMANHO_MATRIX, MPI_INT, worker_id, TAG, MPI_COMM_WORLD);

sleep(1);

int *soma = malloc(sizeof(int) * TAMANHO_MATRIX);
MPI_Recv(soma, TAMANHO_MATRIX, MPI_INT, worker_id, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

for (int i = 0; i < TAMANHO_MATRIX; i++) {
matrixC[linha][i] = soma[i];
}

#ifndef DEBUG
printf("Soma recebida pelo MASTER vindo do worker [%d]: [", worker_id);
imprime_linha(soma);
printf("]\n");
#endif

free(soma);
linha++;
}
}
```


```
printf("\n\n[MASTER]: Matriz C = A + B\n");
imprimir_matrix(matrixC);
printf("\n");
} else {
// estamos no nó escravo
for (int i = 0; i < (TAMANHO_MATRIX / (size - 1)); i++) {
int *linhaA = malloc(sizeof(int) * TAMANHO_MATRIX);
int *linhaB = malloc(sizeof(int) * TAMANHO_MATRIX);
int *soma = malloc(sizeof(int) * TAMANHO_MATRIX);

MPI_Recv(linhaA, TAMANHO_MATRIX, MPI_INT, MASTER, TAG, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
MPI_Recv(linhaB, TAMANHO_MATRIX, MPI_INT, MASTER, TAG, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
// printf("WORKER[%d]: recebeu do mestre:\n", myRank);

#ifndef DEBUG
printf("Linha A: [");
imprime_linha(linhaA);
printf("]\n");

printf("Linha B: [");
imprime_linha(linhaB);
printf("]\n");
#endif

for (int i = 0; i < TAMANHO_MATRIX; i++) {
soma[i] = linhaA[i] + linhaB[i];
}

#ifndef DEBUG
printf("Soma: [");
imprime_linha(soma);
printf("]\n");

printf("\n");
#endif

MPI_Send(soma, TAMANHO_MATRIX, MPI_INT, MASTER, TAG, MPI_COMM_WORLD);

free(linhaA);
free(linhaB);
free(soma);
}
}

MPI_Finalize();
return 0;
}
```

Comentário:

Questão 2

Incorreto

Atingiu 0,00 de 1,00

Analise as afirmativas e, a seguir, marque a alternativa correta:

I - Mecanismos que provêem interoperabilidade entre sistemas distintos pressupõem o uso de um sistema de mensageria e um protocolo de comunicação. Sem esses recursos não há como realizar a interoperabilidade citada

II - No grpc as aplicações que usam protobuf enviam dados em formato binário. Por isso, essas aplicações tem tempo de processamento melhor do que aplicações grpc que fazem uso de formatos como o JSON

III - Num diálogo http/2, se o cliente fizer uma solicitação de recurso para o servidor, este último pode enviar não só o recurso solicitado, mas vários outros associados (sem uma solicitação explícita) na mesma conexão . Essa característica difere o http/2 do http/1.1.

a. Nenhuma das alternativas satisfaz as afirmativas apresentadas

b. Apenas II está correta

c. Apenas II e III estão corretas ✖

d. Apenas I está correta

e. Apenas III está correta

Sua resposta está incorreta.

I - correto. No contexto citado, a interoperabilidade pressupõe um sistema de mensageria e um protocolo de comunicação entre as partes comunicantes.

II - correto.

III - correto.

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

Questão 3

Correto

Atingiu 1,00 de 1,00

Analise as afirmativas e, a seguir, marque a alternativa correta:

I - Apache Kafka é uma plataforma distribuída de tratamento de streaming de eventos em tempo real cujos tópicos podem ser divididos entre vários nós de um cluster

II - No Apache Kafka, os canais de acesso funcionam como uma fila de entrada/saída, no modelo FIFO - First In, First out, onde cada processo sempre recupera o último elemento do canal

III - O mecanismo publish-subscribe do Apache Kafka funciona como um protocolo de comunicação que equaliza os tempos de processamento dos vários consumidores do broker.

a. Apenas as afirmativas I e III estão corretas ✔

b. Apenas as afirmativas I e II estão corretas

c. Nenhuma das opções corresponde às afirmativas apresentadas

d. Todas afirmativas estão corretas

e. Apenas as afirmativas II e III estão corretas

Sua resposta está correta.

I - Correto

II - Incorreto. Os processos podem acessar qualquer elemento da fita de streaming

III - Correto

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

Questão 4

Incorreto

Atingiu 0,00 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. Nenhuma das opções satisfaz as afirmativas apresentadas

c. Apenas as afirmativas I e II estão corretas

d. Apenas a afirmativa III está correta

e. Apenas a afirmativa II está correta

Sua resposta está incorreta.

I - correto

II - correto

III - correto

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

Questão 5

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. Nenhuma das alternativas está correta

c. Apenas a afirmativa II está correta

d. Apenas a afirmativa I está correta ✖

e. Apenas a afirmativa III 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 6

Completo

Atingiu 1,60 de 3,00

Elaborar um microserviço RPC (linguagem C) que 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: (i) arquivo de definição de interface, (ii) os códigos .c do cliente e do servidor da aplicação, e (iii) um README com identificação do aluno (matrícula/nome) e instruções de execução

questao6.zip

Comentário: