questoes - p2

#pspd

- Como rodar arquivos OpenMP:

```
iagocabral@MacBook-Air-de-Iago Provas % gcc-14 -fopenmp -o
divisao_equilibrada 1-Divisao_equilibrada_threads.c

iagocabral@MacBook-Air-de-Iago Provas % OMP_NUM_THREADS=4
./divisao_equilibrada
```

Questao 2

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)
0                           i=i+10;
1           }
2           printf("i=%d\n", i);
3           return 0;
4   }
```

I - A execução com o comando OMP_NUM_THREADS=4 t1 vai imprimir o valor 40
II - Se a linha 9 for suprimida, o binário equivalente acionado com o comando OMP_NUM_THREADS=3 t1 imprimirá sempre o valor 30
III - Se na linha 7 for acrescentada a declaração private(i) e houver supressão da linha 9, o binário equivalente acionado com o
comando OMP_NUM_THREADS=6 t1, o programa vai imprimir 60

Resposta: Todas Falsas

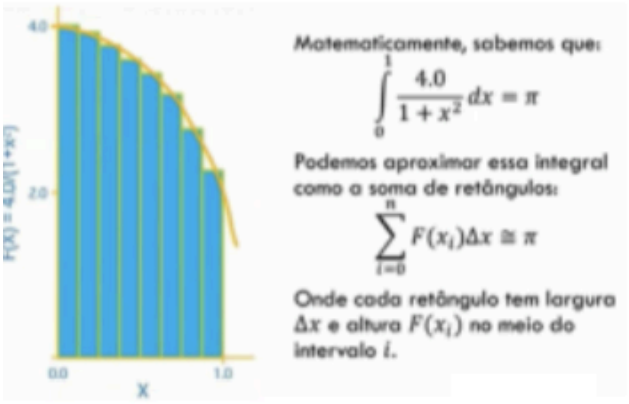
Questao 3

Analise as afirmações a seguir e marque a alternativa correta
I - O problema dos generais bizantinos é uma metáfora que descreve a dificuldade de se entrar em um acordo quando entidades
centralizadas decidem em nome da maioria
II - A tecnologia blockchain é uma solução eficiente para o problema dos generais bizantinos
III - O algoritmo Paxos é uma solução de consenso distribuído cuja variante pode ser usada para coordenação e resolução de impasses
em redes blockchain

Resposta: Apenas II e III estao corretas

Questao 4

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 MPI, de modo a permitir que os processos, em conjunto e colaborativamente, consigam encontrar o valor
de Pi, pela distribuição do cálculo das áreas dos retângulos entre os processos.




```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 */
```

Resposta

```c
#include <stdio.h>

#include <mpi.h>



#define NUM_STEPS 8000000
```

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

int rank, size;

double step, sum = 0.0, partial_sum = 0.0;

double x, pi;



// Inicialização do MPI

MPI_Init(&argc, &argv);

MPI_Comm_rank(MPI_COMM_WORLD, &rank);

MPI_Comm_size(MPI_COMM_WORLD, &size);



// Define o tamanho do passo

step = 1.0 / (double) NUM_STEPS;



// Cada processo realiza parte do cálculo

for (int i = rank; i < NUM_STEPS; i += size) {

x = (i + 0.5) * step;

partial_sum += 4.0 / (1.0 + x * x);

}
```

```
// Reduz as somas parciais de todos os processos no processo 0

MPI_Reduce(&partial_sum, &sum, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);



// O processo 0 calcula o valor final de π e o imprime

if (rank == 0) {

pi = sum * step;

printf("Pi = %.15f\n", pi);

}



// Finaliza o MPI

MPI_Finalize();

return 0;

}
```

Questao 5

```
1   #include <stdio.h>
2   #include <omp.h>
3   int main(){
4       int tid=0, nthreads=0;
5       printf("\nRegião serial (thread única)\n\n");
6       #pragma omp parallel
7       {
8           tid      = omp_get_thread_num();
9           nthreads = omp_get_num_threads();
10          printf("Região paralela (thread %d de %d threads)\n", tid, nthreads);
11      } /*fim-pragma */
12      printf("\nRegião serial (thread única)\n\n");
13      #pragma omp parallel num_threads(4)
14      {
15          tid = omp_get_thread_num();
16          nthreads = omp_get_num_threads();
17          printf("Região paralela (thread %d de %d threads)\n", tid, nthreads);
18      } /* fim-pragma */
19      printf("\nRegião serial (thread única)\n\n");
20      return 0;
21  } /* fim-main */
```

1. Se OMP_NUM_THREADS=6, na segunda região paralela desse código (linhas 13 a 18), serão geradas 10 threads e, portanto, 10
   impressões (linha 17)
2. Se a linha 15 for movida para ficar fora da região paralela (entre as linhas 11 e 13), esse código passa a ser não compilável, pois
   não é possível saber o número de threads em uma região serial do código
3. Esse código é mais apropriado para funcionar em arquiteturas UMA (Uniform Memory Access) ou de memória compartilhada do
   que em arquiteturas NUMA (Non Uniform Memory Access)

Resposta: Apenas a terceira afirmacao esta correta

Questao 6

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.

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

int main(int argc, char *argv[]) {

        int i=0;
        #pragma omp parallel
        {
                if (omp_get_thread_num() == 1)
                        i=i+10;
        }
        printf("i=%d\n", i);
        return 0;
}
```

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

Resposta: 1 e 3 estao corretas

Questao 7

Questao 8

```c
include <stdio.h>
#include <omp.h>
#include <string.h>
#define MAX 100

int main(int argc, char *argv[]) {

        #pragma omp parallel
        {
                int soma=0;
                #pragma omp for
                for (int i=0;i<MAX;i++) {
                  soma +=omp_get_num_threads()*i;
                } /* fim-for */
                printf("Thread[%d] iterou %d vezes\n",
                                omp_get_thread_num(), soma);
        } /* fim omp parallel */
        return 0;
}
```

I - A execução com o comando OMP_NUM_THREADS=4 t1 vai imprimir que cada thread foi executada 25 vezes
II - Se este programa for acionado tendo a variável OMP_NUM_THREADS um valor maior do que o número de núcleos da máquina,
apenas as threads equivalentes ao número de núcleos serão criadas
III - Se o programa for executado numa máquina com 10 núcleos de processamento e a variável OMP_NUM_THREADS estiver com
valor igual a 20, o programa não será ativado

Resposta: todas as afirmativas estao erradas

Questao 9

Por quê os algoritmos de hash consistente são necessários em redes P2P? Que tipo de problemas essa algoritmo resolve? Na resposta,
apresentar um exemplo esclarecendo o funcionamento do algoritmo.

Resposta
Em redes P2P (peer-to-peer), os algoritmos de hash consistente são fundamentais para resolver o problema de distribuição e redirecionamento eficiente de dados entre nós em um sistema descentralizado, especialmente quando nós entram ou saem dinamicamente da rede.
Em um algoritmo de hash consistente, nós e dados são mapeados em um espaço circular de hash (um anel). Cada dado é atribuído ao primeiro nó no sentido horário no

anel, garantindo uma distribuição balanceada. Por exemplo, se temos quatro nós (A, B, C, D) posicionados no anel e chaves (dados) mapeadas entre eles, as chaves são atribuídas ao nó seguinte no anel. Quando um novo nó é adicionado (por exemplo, E), apenas as chaves que ele abrange em sua nova posição são redistribuídas. Da mesma forma, se um nó sair, apenas suas chaves são realocadas para o próximo nó no anel. Isso minimiza a redistribuição de dados, garantindo escalabilidade e eficiência em redes dinâmicas.

Questao 10

```
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

Resposta: nenhuma das opcoes

Questao 11

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.

Respostas: todas corretas

Questao 12

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

Resposta: apenas a 3 esta correta

Questao 13

Julgue as afirmações abaixo e marque a alternativa correta:
I - RDDs (Resilient Distributed Datasets) são estruturas tipadas do Spark que podem ser alteradas por comandos python ou R
II - As transformações (Transformations) são implementadas no Spark em modo lazy, ou seja, são executadas posteriormente, apenas
quando é instanciado uma ação (Action), visando melhoria de performance
III - O Spark é mais rápido do que o Hadoop/Map-Reduce, porque os estágios de execução são implementados com uso intensivo de
memória ao invés de uso de disco (memória secundária).

Resposta correta: II e III

Questao 14

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

Resposta: todas estao corretas

Questao 15

Analise as afirmativas a seguir e marque a alternativa correta.
I - A replicação de dados em diferentes nós melhora a confiabilidade e a performance do sistema como um todo, mas gera problemas
de consistência entre as réplicas.
II - O algoritmo Paxos é uma solução proposta por Lamport a fim de garantir um acordo sobre o estado global do sistema (máquina de
estados distribuída), incluindo valores de variáveis.
III - Uma das soluções de consistência de dados entre réplicas mais eficiente é permitir operações de leitura em apenas um dos nós do
cluster (denominado de master), desde que esse nó faça a atualização dessa operação sofrida para os demais nós do cluster de
replicação.

Resposta: I e II estao corretas