CONCURSO ITA 2025

EDITAL: 03/ITA/2025

CARGO: TECNOLOGISTA

PERFIL: TL-05

CADERNO DE QUESTÕES

1. Esta prova tem duração de **4 (quatro) horas**.
2. Você poderá usar **apenas** caneta esferográfica de corpo transparente com tinta preta, lápis ou lapiseira, borracha, régua transparente simples e compasso. **É proibido portar qualquer outro material escolar ou equipamento eletrônico.**
3. Esta prova é composta de **25 questões de múltipla escolha** (numeradas de 01 a 25) e de **3 questões dissertativas**.
4. Você recebeu este **caderno de questões, uma folha de leitura óptica e um caderno de respostas** que deverão ser devolvidos ao final do exame.
5. As questões de **múltipla escolha devem ser respondidas na folha de leitura óptica**. Assinale a opção correspondente à resposta de cada uma das questões, de **01 a 25**. Cada questão de múltipla escolha admite uma única resposta.
6. A folha de leitura óptica, deve ser preenchida usando caneta preta. Você deve preencher todo o campo disponível para a resposta, sem extrapolar os limites, conforme instruções na folha de leitura óptica.
7. Cuidado para não errar no preenchimento da folha de leitura óptica. Ela não será substituída.
8. Não haverá tempo suplementar para o preenchimento da folha de leitura óptica.
9. As **questões dissertativas devem ser respondidas no caderno de respostas. Responda usando caneta preta, no campo destinado a cada questão**.
10. **É obrigatória a devolução do caderno de questões, do caderno de respostas e da folha de leitura óptica**, sob pena de desclassificação do candidato.
11. **Aguarde o aviso para iniciar a prova. Ao terminá-la, avise o fiscal e aguarde-o no seu lugar**.

Questão 1. Suponha um programa cujo fração paralelizável é 80% (0,8). Qual é o speedup esperado ao rodá-lo em 8 processadores idênticos segundo a Lei de Amdahl?

A ( ) 2,50

B ( ) 3,33

C ( ) 4,00

D ( ) 5,00

E ( ) 6,67

Questão 2. Usando o modelo de Gustafson, se a fração serial do problema é 10% e aumentamos a escala de trabalho para 100 processadores, qual o speedup escalado aproximado?

A ( ) 10,0

B ( ) 50,0

C ( ) 90,1

D ( ) 99,0

E ( ) 100,0

Questão 3. Se um programa roda em 64 processos com speedup observado de 40, qual é a eficiência percentual?

A ( ) 40%

B ( ) 62.5%

C ( ) 64%

D ( ) 156%

E ( ) 160%

Questão 4. O padrão MPI garante que mensagens enviadas de um mesmo processo A para um mesmo processo B (no mesmo comunicador) com a mesma tag sejam lidas no receptor na mesma ordem em que foram enviadas?

A ( ) Não, a ordenação só é garantida com tags diferentes

B ( ) Não — MPI não garante ordem alguma entre mensagens.

C ( ) Sim, mas somente se for usado MPI_Barrier antes dos envios.

D ( ) Apenas se usar send/recv bloqueante; com non-blocking a ordem não é garantida.

B ( ) Sim — MPI garante ordenação entre pares (sender, receiver) no mesmo comunicador.

Questão 5. Em OpenMP, ao usar #pragma omp parallel for reduction(+:sum) sobre um loop, qual das afirmações é verdadeira sobre sum durante a execução paralela?

A ( ) sum só é consistente se o schedule for static.

B ( ) sum não pode ser usada dentro de reductions; OpenMP exige locks.

C ( ) sum torna-se uma variável threadprivate persistente entre regiões paralelas.

D ( ) sum é private para cada thread e ao final as cópias são combinadas (reduzidas).

E ( ) sum é compartilhada por todas as threads e cada atualização é atômica por padrão.

Questão 6. Qual afirmação melhor descreve false sharing?

A ( ) Ocorre apenas em arquiteturas NUMA remotas.

B ( ) É um problema de link editing entre bibliotecas compartilhadas.

C ( ) É resolvido automaticamente por compiladores que aplicam padding.

D ( ) Ocorre quando variáveis frequentemente atualizadas por threads distintas residem na mesma linha de cache.

E ( ) Ocorre quando variáveis compartilhadas por threads distintas são armazenadas em registradores, não visualizando alterações na memória principal.

Questão 7. Um sistema tem pico de 200 GFLOP/s e largura de banda de memória de 50 GB/s. Se uma aplicação tem intensidade operacional de 1 FLOP/byte, qual o teto (teórico) de desempenho FLOP/s segundo o modelo roofline?

A ( ) 200 GFLOP/s

B ( ) 100 GFLOP/s

C ( ) 50 GFLOP/s

D ( ) 25 GFLOP/s

E ( ) 4 GFLOP/s

Questão 8. Qual das alternativas abaixo melhor descreve divergência de warp (warp divergence) em GPUs?

A ( ) Threads de um mesmo warp acessando memória global de forma não-coalescente, aumentando a taxa de cache misses.

B ( ) Branch condicional com threads do mesmo warp tomando caminhos diferentes de execução.

C ( ) Threads de um warp executando operações de ponto flutuante de precisão diferente.

D ( ) Threads de um warp aguardam para sempre em uma sincronização, quando ao menos uma thread deste warp salta o sincronismo.

E ( ) Quando warps consecutivos são alocados para serem executados em SMs diferentes.

Questão 9. Comparando um único dispositivo HDD magnético com um único NVMe SSD em cenários práticos de HPC, qual é a melhor descrição típica?

A ( ) SSDs são sempre piores em throughput sequencial.

B ( ) HDD tem maior IOPS aleatória; SSD tem maior latência.

C ( ) HDD tem baixa latência; SSD tem alta banda sequencial.

D ( ) Ambos têm desempenho idêntico em acessos sequenciais.

E ( ) HDD tem boa largura de banda sequencial; SSD (NVMe) tem alta IOPS aleatória e menor latência.

Questão 10. Qual é a propriedade principal do RDMA (Remote Direct Memory Access) usada em InfiniBand para HPC?

A ( ) Copia dados via kernel do SO remoto, envolvendo o CPU remoto para cópia.

B ( ) Permite que um processo no host remoto modifique o disco do outro nó diretamente.

C ( ) Permite operações one-sided sem prévia inscrição (registration) da memória remota ou estabelecimento de permissões no nó alvo.

D ( ) Permite leitura/escrita direta da memória do nó remoto sem intervenção do CPU remoto (one-sided), reduzindo latência e sobrecarga de CPU.

E ( ) Fornece coerência de cache entre CPUs de hosts distintos, de modo que leituras RDMA reflitam imediatamente escritas feitas por CPUs remotos.

Questão 11. No Slurm, qual a diferença típica entre sbatch e srun?

A ( ) sbatch cancela jobs; srun monitora recursos.

B ( ) srun submete job em lote; sbatch só exibe fila.

C ( ) Ambos fazem a mesma coisa; srun é alias de sbatch.

D ( ) srun só é para nós de login, sbatch só para nós de computação.

E ( ) sbatch submete job em lote; srun é usado para executar tarefas interativas/etapas dentro do job.

Questão 12. Considere o loop (em linguagem C):

```
for (int i = 1; i < n; ++i) {
  A[i] = A[i] + A[i-1];
}
```

Que tipo de dependência existe que impede paralelização direta por iteração?

A ( ) Dependência atômica obrigatória.

B ( ) Dependência RAW (read after write).

C ( ) Dependência WAR (write after read).

D ( ) Dependência WAW (write after write).

E ( ) Não há dependência entre iterações — loop é totalmente paralelizável.

Questão 13. Em sistemas HPC, qual característica descreve corretamente sistemas como Lustre?

A ( ) Suportam acesso POSIX.

B ( ) São projetados apenas para arquivos pequenos (<4 KB).

C ( ) Não permitem espaço de nome compartilhado entre nós.

D ( ) Armazenamento centralizado com metadados e dados ambos num único disco.

E ( ) Distribuem dados entre múltiplos servidores (OSTs), em detrimento de throughput agregado alto devido ao striping.

Questão 14. Num sistema HPC, um crescimento súbito na taxa de page faults e aumento de swap indica, tipicamente:

A ( ) Rede congestionada.

B ( ) Bom uso de cache L1.

C ( ) CPU bound — não relacionado à memória.

D ( ) Problema apenas no sistema de arquivos paralelo.

E ( ) Memory thrashing / falta de memória física; possivelmente configuração de job/quotas incorreta.

Questão 15. Em VMs na nuvem voltadas a HPC, qual técnica é usada para obter comunicação de baixa latência e alto throughput em aplicações que exigem RDMA?

A ( ) Drivers paravirtualizados (virtio-net) sobre TCP/IP.

B ( ) SR-IOV ou PCI passthrough de interfaces InfiniBand/Ethernet.

C ( ) Armazenamento local efêmero para diminuir dependência de rede.

D ( ) Overlay networks como VXLAN/GRE para encapsular tráfego MPI.

E ( ) NUMA-aware scheduling para reduzir impacto da migração de páginas.

Questão 16. Em arquiteturas modernas de HPC que combinam CPU multinúcleo e aceleradores GPU, o gargalo de desempenho muitas vezes ocorre na transferência de dados. Qual prática é fundamental para minimizar a latência nesse cenário?

A ( ) Utilizar somente comunicação MPI síncrona entre CPU e GPU.

B ( ) Desativar cache L3 para reduzir a interferência entre threads e GPU.

C ( ) Consolidar múltiplas GPUs em um único domínio de memória virtual sem NUMA binding.

D ( ) Forçar escalonamento round-robin entre processos MPI para maximizar uniformidade.

E ( ) Utilização de pinned memory e comunicação assíncrona para sobreposição de computação e I/O.

Questão 17. Qual das seguintes afirmações melhor descreve o conceito de "warps" no contexto da execução de programas CUDA em GPUs?

A ( ) Um warp é um conjunto que contém 32 threads, os quais são executados em paralelo em uma GPU CUDA.

B ( ) Um warp é uma coleção de threads que são executadas sequencialmente para minimizar o uso de recursos da GPU.

C ( ) Warp é um algoritmo de otimização automática de código CUDA que visa reduzir a complexidade do paralelismo em GPUs.

D ( ) Warp é um tipo de memória especializada dentro da GPU que armazena temporariamente dados para kernels CUDA.

E ( ) Warp é um mecanismo de comunicação inter-kernel utilizado para sincronizar diferentes kernels CUDA durante a execução.

Questão 18. Considere o uso de diretivas OpenMP em um loop paralelizado. Qual das seguintes diretivas é mais apropriada para garantir que uma seção crítica de código, que atualiza uma variável compartilhada, seja executada por apenas um thread de cada vez, sem comprometer o paralelismo do restante do loop?

A ( ) #pragma omp barrier

B ( ) #pragma omp critical

C ( ) #pragma omp master

D ( ) #pragma omp ordered

E ( ) #pragma omp parallel for

Questão 19. Considere um ambiente de supercomputação gerenciado pelo SLURM (Simple Linux Utility for Resource Management). Qual das seguintes opções melhor descreve a função da diretiva #SBATCH --ntasks-per-node em um script de job SLURM e como ela pode influenciar a alocação de recursos e o desempenho de uma aplicação paralela?

A ( ) Configura o número de nós a serem utilizados por job, impactando a escala do paralelismo distribuído.

B ( ) Define o número total de jobs que podem ser submetidos por nó, limitando a carga de trabalho para evitar sobrecarga.

C ( ) Aloca o número de threads por nó para aplicações multithreaded, afetando diretamente a configuração de OpenMP.

D ( ) Determina o número de partições disponíveis para o job, otimizando o acesso a diferentes tipos de recursos computacionais.

E ( ) Especifica o número de tarefas (processos) a serem executadas por nó, o que pode ser usado para controlar o grau de paralelismo e a distribuição de carga em um nó.

Questão 20. No contexto de um cluster gerenciado pelo SLURM (Simple Linux Utility for Resource Management), qual é a função do usuário slurm e por que ele é importante para a segurança e a administração do sistema?

A ( ) O usuário slurm é utilizado para fazer login e submeter jobs diretamente ao sistema de filas.

B ( ) O usuário slurm é utilizado exclusivamente para monitorar o uso de recursos e gerar relatórios de performance do cluster.

C ( ) O usuário slurm é responsável por executar jobs dos usuários, garantindo que cada job tenha os privilégios aumentados acima do usuário para garantir sua execução.

D ( ) O usuário slurm é um administrador do sistema com plenos privilégios para gerenciar todos os aspectos do cluster, incluindo a instalação de software e atualizações de segurança.

E ( ) O usuário slurm é o proprietário dos processos do daemon SLURM, garantindo que o gerenciamento de recursos e a execução de jobs sejam realizados com privilégios reduzidos aumentando a segurança.

Questão 21. A respeito das operações de comunicação coletiva, como MPI_Scatter e MPI_Gather, apresentadas a seguir, indique a alternativa INCORRETA:

```
int MPI_Scatter(const void *sendbuf, int sendcount, MPI_Datatype
                sendtype, void *recvbuf, int recvcount,
                MPI_Datatype recvtype, int root, MPI_Comm comm)

int MPI_Gather(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
               void *recvbuf, int recvcount, MPI_Datatype recvtype,
               int root, MPI_Comm comm)
```

A ( ) Nessas operações, o papel de cada processo da aplicação é determinado pelo rank dentro do grupo indicado.

B ( ) Operações de comunicação coletiva em MPI envolvem todos os processos de um dado comunicador (MPI_Comm), especificado como parâmetro na chamada.

C ( ) Cabe à biblioteca MPI elaborar mecanismos de controle dos processos participantes e de suas localizações nos nós (computadores) utilizados pela aplicação.

D ( ) O número de processos existentes no grupo é tratado pelo programador no particionamento dos dados, indicando os limites dos fragmentos de mensagem em cada operação.

E ( ) O número de mensagens necessárias para realizar as operações de transmissão de dados é dependente da tecnologia de rede utilizada, mas isso é tratado pela biblioteca, sem necessidade de ação específica do programador.

Questão 22. Considere o trecho de código C para multiplicação de 2 matrizes apresentado a seguir, instrumentado com diretivas OpenMP, e assinale a alternativa INCORRETA:

```
     …
     int i, j, k;
     …
1    // #pragma omp parallel for private(j,k)
2    for (i=0; i<N; i++)
3      // #pragma omp parallel for private(k)
4      for (j=0; j<N; j++) {
5        soma = 0;
6        // #pragma omp parallel for reduction(+:soma)
7        for (k=0; k<N; k++)
8          soma += A[i][k] * B[k][j];
9        C[i][j] = soma;
10     }
```

A ( ) A utilização da diretiva comentada na linha 1 geraria a criação de um time de threads que dividiria o cálculo das linhas da matriz C.

B ( ) A utilização da diretiva comentada na linha 3 geraria a criação de um time de threads que dividiria o cálculo das colunas de cada linha da matriz C.

C ( ) A utilização da diretiva comentada na linha 6 geraria a criação de um time de threads que realizaria somas parciais para o cálculo de cada elemento da matriz C.

D ( ) O uso da cláusula “private(j,k)” na diretiva comentada na linha 1 seria desnecessário, pois a paralelização neste caso ocorreria sobre linhas independentes.

E ( ) A cláusula “reduction(+:soma)” na diretiva comentada na linha 6 geraria a criação de uma cópia da variável “soma” para cada thread do time criado, inicializada com valor 0 e reduzida pela operação de soma ao final da região paralela.

Questão 23. Na programação com aceleradores, usando CUDA, por exemplo, é comum que se utilize um modelo SIMT, em que os núcleos de processamento do acelerador sejam usados para executar o mesmo código sobre partes distintas dos dados. Considerando o trecho de código a seguir, que é parte de uma soma de vetores, indique a alternativa CORRETA:

```
#define N (1<<20)
…
__global__
void add(int n, float *x, float *y) {
  int index = blockIdx.x * blockDim.x + threadIdx.x;
  int stride = blockDim.x * gridDim.x;

  for (int i = index; i < n; i += stride)
    y[i] = x[i] + y[i];
}
…
main() {
  …
  int blockSize = 256;
  int numBlocks = (N + blockSize - 1) / blockSize;

  add <<< numBlocks, blockSize >>> (N, x, y);
  …
}
```

A ( ) Os índices das threads, passados como parâmetros na invocação de um kernel, indicam qual elemento cada thread deve manipular.

B ( ) A organização da grade de blocos de threads, realizada na invocação de um kernel, fixa a execução das threads aos blocos multiprocessadores (SMs) da GPU.

C ( ) A organização das grades de blocos de threads para execução pode ser realizada em uma, duas ou três dimensões, que têm igual impacto no desempenho do programa.

D ( ) Pelo cálculo dos índices das threads, é possível que existam threads que não devem realizar a manipulação dos dados, pois extrapolariam o número de elementos nas estruturas particionadas.

E ( ) Cada thread será responsável pelo cálculo de 1 elemento do vetor resultante, associado ao número do bloco multiprocessador (Stream Multiprocessor - SM) em que essa thread é executada.

Questão 24. Considere o trecho de código C a seguir, instrumentado e compilado com com recursos das extensões OpenMP, e assinale a alternativa INCORRETA:

```
double dotProd(double *a, int *b, long int N) {
  long int i;
  double dot = 0.0;

  #pragma omp parallel for simd reduction(+:dot)
  for(i = 0; i < N; i++)
    ot += a[i] * b[i];

  return dot;
}
```

A ( ) A cláusula de redução gerará a criação de uma cópia privada da variável dot para cada thread da região paralela.

B ( ) O cálculo do produto escalar interno (dot) será realizado dividindo-se as iterações do loop for entre múltiplas threads.

C ( ) As iterações do loop for podem beneficiar-se de instruções SIMD do processador, realizando a mesma operação aritmética em mais de um elemento de cada vetor de uma vez.

D ( ) O operador simd na diretiva parallel torna o código gerado sujeito a problemas de perda de desempenho, devido à manipulação de posições contíguas da memória pelas diferentes threads.

E ( ) A variável dot não será compartilhada entre as threads da região paralela e, por isso, não é preciso usar mecanismos explícitos de exclusão mútua na atualização do valor antes do retorno da função.

Questão 25. Na programação paralela com aceleradores, como em CUDA, é preciso considerar os espaços de endereçamento acessíveis pelo código executando em CPU e pelo código executado em GPU.

```
// código 1:

  int N = 1<<20;
  float *x, *y, *d_x, *d_y;

  x = (float*)malloc(N*sizeof(float));
  y = (float*)malloc(N*sizeof(float));

  cudaMalloc(&d_x, N*sizeof(float));
  cudaMalloc(&d_y, N*sizeof(float));

  // inicia elementos dos vetores x e y
  cudaMemcpy(d_x, x, N*sizeof(float), cudaMemcpyHostToDevice);
  cudaMemcpy(d_y, y, N*sizeof(float), cudaMemcpyHostToDevice);

  // invoca kernel para manipulação de x e y
  kern <<< ..., ... >>> (N, d_x, d_y);

  cudaMemcpy(y, d_y, N*sizeof(float), cudaMemcpyDeviceToHost);

// código 2:

  int N = 1<<20;
  float *x, *y;

  cudaMallocManaged(&x, N*sizeof(float));
  cudaMallocManaged(&y, N*sizeof(float));

  // inicia elementos dos vetores x e y
  …

  // invoca kernel para manipulação de x e y
  kern <<< ..., ... >>> (N, x, y);

  cudaDeviceSynchronize();
```

Observe os 2 trechos de código apresentados anteriormente e indique a alternativa INCORRETA (próxima página):

A ( ) No código 1, vê-se a cópia de dados da memória RAM do computador para alguma área de memória da GPU.

B ( ) No código 1, por tratarem-se de áreas de memória separadas, dados alterados na memória da GPU precisam ser copiados de volta para a RAM, caso sejam necessários após seus processamentos em GPU.

C ( ) No código 1, observa-se a alocação de áreas de memória separadas nos espaços de endereçamento da(s) CPU(s) e da GPU, com atribuição de valores na área de GPU diretamente pelo código em CPU.

D ( ) No código 2, vê-se que uma operação de sincronização pode ser necessária para garantir a consistência dos dados a serem manipulados em CPU após suas alterações em GPU, embora cópias explícitas não sejam necessárias.

E ( ) No código 2, vê-se a alocação de memória de forma unificada, sendo que um mesmo endereço para área de memória usado no código em CPU pode também ser usado pelo código em GPU, sem cópias explícitas no programa.

Questão discursiva 1. Discorra sobre gerenciadores de pacotes de softwares ou módulos de aplicativos em ambiente compartilhado e uso de containers. Cite exemplos de sistemas gerenciadores e suas funcionalidades.

Questão discursiva 2. Discorra sobre o monitoramento e gerenciamento de desempenho de um sistema computacional de alto desempenho, relacionado a CPU, memória, redes de dados, sistemas de arquivos.

Questão discursiva 3. Discorra sobre a identificação de “gargalos”/fatores limitantes de desempenho, relacionados a processamento numérico em CPU, uso de memória e dispositivos de entrada/saída, e suas possíveis estratégias de mitigação.

RASCUNHO

RASCUNHO

RASCUNHO

RASCUNHO