AVALIAÇÃO FINAL – SUPERCOMPUTAÇÃO

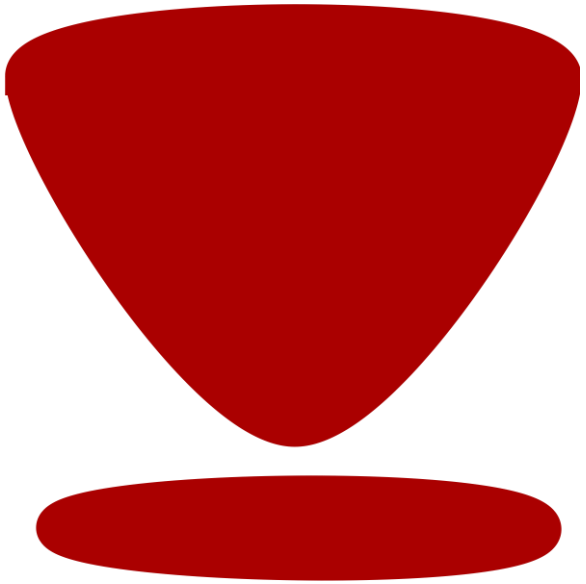
- Bem-vindo(a)s a prova final da disciplina de Supercomputação.
- Essa prova consiste de 4 questões. O tempo total de prova é de 2 horas, e cada questão deve ser respondida em até 30 minutos.
- A prova é síncrona. Isto é, todos mudam de questão simultaneamente. Todavia, nada impede que você continue a resolver a questão anterior, mas atente-se ao tempo.
- Com exceção da questão 1, não estão permitidas conversas entre os membros do grupo. **Após a questão 1, a prova segue individual**. O administrador do cluster só possui papel efetivo na primeira questão. Nas demais questões, a responsabilidade da execução é de cada aluno.
- Está permitida consulta ao site da disciplina e documentação oficial das bibliotecas utilizadas.
- Na sua mesa, há alguns materiais impressos. Consulte-os.
- **Você receberá uma folha de instruções individual. Essa folha deve ser preenchida com o seu nome e devolvida ao final da prova.**
- Enquanto não começa a prova, aproveite para garantir que o cluster está em pleno funcionamento: partes montadas, máquinas virtuais ligadas, filas aptas para recebimento de Jobs. Um teste de sanidade é pedir uma sessão iterativa (e recebê-la) e/ou submeter um job batch “hello world” conforme visto em sala de aula.
- Alunos atrasados serão admitidos ainda na questão 01. Após a questão 01, devem realizar prova substitutiva.
- Local da prova: Laboratório Ágil – Horários: abertura das salas (09h), início da prova (10h), final da prova (12h).
- BOA PROVA!

QUALQUER CONSULTA EM MATERIAL NÃO AUTORIZADO, IMPLICA EM NOTA ZERO NA PROVA.

Insper

A prova vai começar!

Faça download da prova



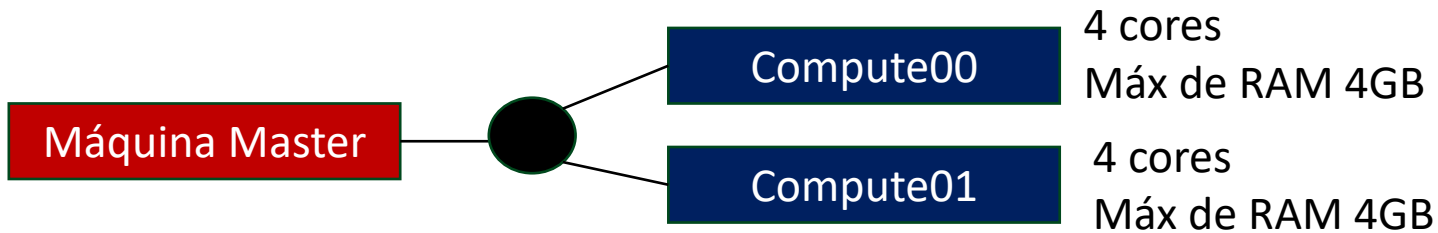
- Atenção:
- A prova deve ser resolvida NO CLUSTER, com seu usuário.
- Faça download com

git clone https://github.com/andrefmb/i_p2

Questão 01 - SLURM

Esta questão é voltada para a customização do Cluster para execução da prova. Ela é dividida em questões que devem ser resolvidas pelo administrador do cluster e questões que devem ser resolvidas pelos demais membros.

O administrador deve configurar o cluster para a seguinte topologia:



Além disso, deve ser criada uma PARTIÇÃO chamada PROVA, com walltime máximo de 20 minutos, que permita tratar os recursos computacionais como CPU e memória como consumíveis e compartilhados. Altere para que a partição PROVA seja a partição PADRÃO. Nessa partição estarão os dois nós (compute00 e compute01).

ATENÇÃO: A partir dessa configuração, todas as questões devem ser executadas na partição PROVA.

Demais integrantes: há na folha de instrução sua tarefa específica para essa questão.

Questão 02 - MPI

Situação Problema – Buscando por um valor em um grande array

Precisamos implementar um programa em MPI que permita a busca por um valor específico em um grande array. Nesse programa, o processo de rank 0 deve gerar um array de tamanho aleatório* e popular esse vetor aleatoriamente. Após isso, ele deve escolher* um número a ser buscado no vetor. A busca deve ser realizada por todos os processos MPI.

O processamento paralelo deve se dar da seguinte forma:

- Cada processo com rank <> 0 deve receber a porção do array que lhe cabe, assim como qual é o número que está sendo buscado
- Cada processo com rank <> 0 deve buscar o número procurado no array e quando encontrar encaminhar ao processo 0 o índice do vetor em que encontrou o elemento.
- Devem ser enviadas todas as ocorrências do elemento procurado no vetor

- Ao final, o processo de rank 0 deve exibir uma lista de todos os índices em que o número se encontra.

SUA TAREFA:

1. Faça o download da solução serial:
git clone https://github.com/andrefmb/i_p2

2. Implemente a solução usando MPI_SEND e MPI_RECV (ou MPI_SCATTER) [1,5 pontos]

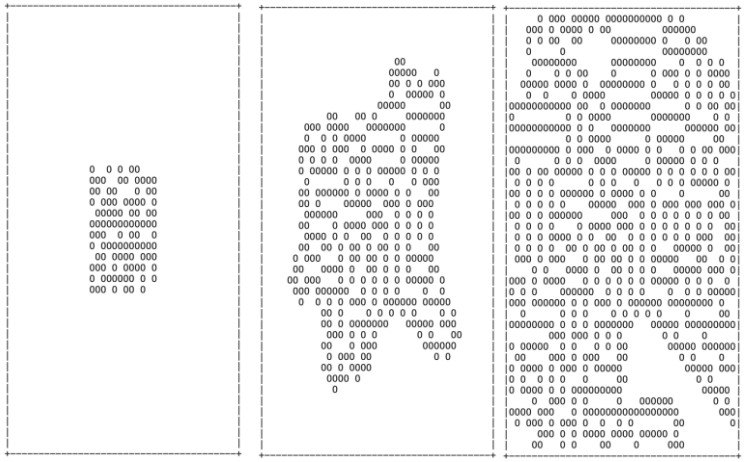
3. Modifique a implementação anterior, de modo que não é mais necessário enviar todas as ocorrências, mas apenas a quantidade de ocorrências [0,5 ponto]

4. Modifique a implementação anterior, de modo que o rank 0 deve receber a quantidade de ocorrências com MPI_REDUCE. [0,5 ponto]

ATENTE-SE AO FATO DE QUE NA SUA FOLHA FÍSICA HÁ INSTRUÇÕES ESPECÍFICAS PARA ESSA RESOLUÇÃO. OS ITENS COM * REPRESENTAM ITENS QUE POSSUEM INSTRUÇÕES ESPECÍFICAS.

QUALQUER CONSULTA EM MATERIAL NÃO AUTORIZADO, IMPLICA EM NOTA ZERO NA PROVA.

Questão 03 - OpenMP



Autômato Celular: Considere um autômato celular em um grid *n x n*, com células indexadas como (i,j). Cada célula pode estar viva ou vazia. Seja $N_{i,j}$ o número de vizinhos de (i,j). Nesse caso, 0 <= $N_{i,j}$ <= 8.
Em uma geração, todas as células são simultaneamente atualizadas de acordo com as seguintes regras:

- Uma célula viva em (i,j) sobrevive se $N_{i,j} \in S_{survive}$
- Uma célula vazia em (i,j) se torna viva se $N_{i,j} \in S_{birth}$

Você recebeu o código-fonte base para essa implementação. Siga as instruções de sua folha para conhecer os valores iniciais de $S_{survive}$ e S_{birth}. Observe que seu código possui uma função que inicializa uma porção do autômato como vivo. Na sua folha de instruções há o número de geração que você deve simular.

SUA TAREFA:

1. Faça o download da solução serial:

git clone https://github.com/andrefmb/i_p2. Modifique algumas linhas do código-fonte conforme sua folha de instruções.

2. Adapte o código para que as simulações sejam processadas por threads OpenMP. Você deve simular tamanhos diferentes de grid (conforme sua folha de instruções), em função de um número de threads, também especificado na sua folha de instruções. Para cada par (tamanho do grid, nro de threads) calcule o walltime de execução. [1,5 pontos]. Na adaptação, inclua obrigatoriamente o código disponível no arquivo incluir_omp.cpp

3. Explique qual foi seu raciocínio para dividir esse trabalho com threads. Comente sobre o speed up. [1 ponto]

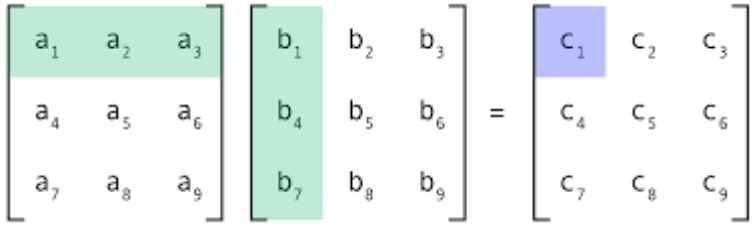
Na mudança para OpenMP, você não precisa executar a função printGrid().

ATENTE-SE AO FATO DE QUE NA SUA FOLHA FÍSICA HÁ INSTRUÇÕES ESPECÍFICAS PARA ESSA RESOLUÇÃO.

QUALQUER CONSULTA EM MATERIAL NÃO AUTORIZADO, IMPLICA EM NOTA ZERO NA PROVA.

Insper

Questão 04 – GPU / THRUST



Max Pooling em CNNs: CNN é um tipo de arquitetura de rede neural projetada para processar dados matriciais, como imagens e vídeos. Ela é amplamente utilizada em tarefas de visão computacional, reconhecimento de padrões e análise de imagens.

Nas CNNs a operação fundamental é a convolução, que aplica um filtro sobre uma região da imagem condensando as características locais.

Em termos práticos, a convolução é a multiplicação de 2 matrizes, A e B, representando, respectivamente, a matriz de pixels e o filtro em questão (pesos da rede).

Obviamente, as matrizes A e B precisam ter tamanhos compatíveis para permitir a multiplicação. Tal multiplicação pode gerar representações de alta dimensão, pois o filtro se desliza sobre a imagem (matriz de pixels).

Para reduzir a dimensionalidade espacial das características (matriz resultante das multiplicações), aplica-se sobre ela outra operação de condensação de features, geralmente, um max pooling, que retorna o maior valor daquela região. Max pooling preserva as características mais proeminentes, ajudando na detecção de padrões.

SUA TAREFA:

1. Faça uma cópia do Colab com a solução parcial. Entenda o que o código C++ está fazendo. Link: https://encurtador.com.br/fkRV9

2. Usando o código Python do Colab, gere matrizes quadradas como input/output, variando o tamanho de N.

3. Complete o código em C++ para realizar a multiplicação de matrizes quadradas, *usando a GPU*, retornando o resultado da aplicação do max pooling sobre ela toda (ou seja, para uma matriz NxN a resposta será apenas 1 float). [1 ponto]

4. Compile e rode seu código completo em C++ obtendo o tempo necessário para rodar o algoritmo com tamanhos diferentes de N. Deixe claro a quantidade de tempo de cada execução em função de N. [0.5 ponto]

5. Explique em uma célula markdown se os tempos estão de acordo com sua expectativa e quais outras alterações da lógica inicial poderiam acelerar a execução na GPU [1 ponto]

Atenção aos itens que devem ser personalizados:

A

Questão 01:

Se você for o administrador, siga as instruções do slide. Caso contrário:

Prepare um job para executar na fila PROVA assim que ela estiver disponível. O seu job deve executar na topologia MPI descrita na caixa "Topologia" (abaixo), e deve ser um programa C++ que obtém o hostname da máquina e imprime o nome da máquina em que executou.

Questão 02:

1. Tamanho aleatório do array: deve ser de pelo menos 10.000 elementos

1. A escolha do número: seu array é de tamanho n. O número escolhido pelo processo de rank 0 deve ser o elemento da posição n/2

Questão 03:

O tamanho inicial do seu grid deve ser de 60 x 60. (versão serial)
$S_{survive}$ = { 3, 4, 5}
S_{birth} = {3}

Número de gerações para executar: 150

Na implementação com OpenMP:
Você deve simular tamanhos de grids (nxn) = { 64, 96, 128} e número de threads = { 2, 3 , 4}

TOPOLOGIA DE RESOLUÇÃO DAS QUESTÕES (via SLURM)
Nas questões de OpenMP seu programa deve sempre executar obrigatoriamente no nó compute00
Nas questões de MPI, a topologia de toda a sua resolução será 3 processos, obrigatoriamente nos mesmos nó de processo.
A tag MPI na questão 02 deve ser 20. Atenção: você pode criar outras TAGs para resolver o problema. Basta que uma delas seja esse valor.

As questões 02 e 03 devem ser submetidas sempre via SLURM. Seu job obrigatoriamente deve receber um nome da seguinte forma:
PROVA_A_<PRIMEIRO_NOME><ULTIMO_NOME> fazendo a substituição das tags por seus primeiro e último nomes, respectivamente.
Todos os Jobs devem, obrigatoriamente, executar na partição PROVA. Caso seu job não seja executado nessa partição, haverá decréscimo de 0,5 ponto por questão.

Resoluções incompletas serão consideradas para fins de correção, mas podem no máximo valor 50% da questão em si.
A resolução completa, porém sem submissão no cluster, também serão corrigidas e pontuadas em até 50% da questão.

Atenção aos itens que devem ser personalizados:

B

Questão 01:

Se você for o administrador, siga as instruções do slide. Caso contrário:

Prepare um job para executar na fila PROVA assim que ela estiver disponível. O seu job deve executar na topologia MPI descrita na caixa "Topologia" (abaixo), e deve ser um programa C++ que obtém o hostname da máquina e imprime o nome da máquina em que executou.

Questão 02:

1. Tamanho aleatório do array: deve ser de pelo menos 7.000 elementos

1. A escolha do número: seu array é de tamanho n. O número escolhido pelo processo de rank 0 deve ser o elemento da posição n/3

Questão 03:

O tamanho inicial do seu grid deve ser de 60 x 60. (versão serial)
$S_{survive}$ = { 4, 5, 6}
S_{birth} = {4}

Número de gerações para executar: 200

Na implementação com OpenMP:
Você deve simular tamanhos de grids (nxn) = { 96, 128} e número de threads = { 2, 3 , 4}

TOPOLOGIA DE RESOLUÇÃO DAS QUESTÕES (via SLURM)
Nas questões de OpenMP seu programa deve sempre executar obrigatoriamente no nó compute00
Nas questões de MPI, a topologia de toda a sua resolução será 4 processos.
A tag MPI na questão 02 deve ser 40. Atenção: você pode criar outras TAGs para resolver o problema. Basta que uma delas seja esse valor.

As questões 02 e 03 devem ser submetidas sempre via SLURM. Seu job obrigatoriamente deve receber um nome da seguinte forma:
PROVA_B_<PRIMEIRO_NOME><ULTIMO_NOME> fazendo a substituição das tags por seus primeiro e último nomes, respectivamente.
Todos os Jobs devem, obrigatoriamente, executar na partição PROVA. Caso seu job não seja executado nessa partição, haverá decréscimo de 0,5 ponto por questão.

Resoluções incompletas serão consideradas para fins de correção, mas podem no máximo valor 50% da questão em si.
A resolução completa, porém sem submissão no cluster, também serão corrigidas e pontuadas em até 50% da questão.

C

Atenção aos itens que devem ser personalizados:

Questão 01:

Se você for o administrador, siga as instruções do slide. Caso contrário:

Prepare um job para executar na fila PROVA assim que ela estiver disponível. O seu job deve executar na topologia MPI descrita na caixa "Topologia" (abaixo), e deve ser um programa C++ que obtém o hostname da máquina e imprime o nome da máquina em que executou.

Questão 02:

1. Tamanho aleatório do array: deve ser de pelo menos 5.000 elementos

1. A escolha do número: seu array é de tamanho n. O número escolhido pelo processo de rank 0 deve ser o elemento da posição n/4

Questão 03:

O tamanho inicial do seu grid deve ser de 80 x 80. (versão serial)
$S_{survive}$ = { 5, 6, 7}
S_{birth} = {3, 4}

Número de gerações para executar: 150

Na implementação com OpenMP:
Você deve simular tamanhos de grids (nxn) = { 64, 96} e número de threads = { 2, 3}

TOPOLOGIA DE RESOLUÇÃO DAS QUESTÕES (via SLURM)
Nas questões de OpenMP seu programa deve sempre executar obrigatoriamente no nó compute01
Nas questões de MPI, a topologia de toda a sua resolução será 5 processos.
A tag MPI na questão 02 deve ser 30. Atenção: você pode criar outras TAGs para resolver o problema. Basta que uma delas seja esse valor.

As questões 02 e 03 devem ser submetidas sempre via SLURM. Seu job obrigatoriamente deve receber um nome da seguinte forma:
PROVA_C_<PRIMEIRO_NOME><ULTIMO_NOME> fazendo a substituição das tags por seus primeiro e último nomes, respectivamente.
Todos os Jobs devem, obrigatoriamente, executar na partição PROVA. Caso seu job não seja executado nessa partição, haverá decréscimo de 0,5 ponto por questão.

Resoluções incompletas serão consideradas para fins de correção, mas podem no máximo valor 50% da questão em si.
A resolução completa, porém sem submissão no cluster, também serão corrigidas e pontuadas em até 50% da questão.

Atenção aos itens que devem ser personalizados:

D

Questão 01:

Se você for o administrador, siga as instruções do slide. Caso contrário:

Prepare um job para executar na fila PROVA assim que ela estiver disponível. O seu job deve executar na topologia MPI descrita na caixa "Topologia" (abaixo), e deve ser um programa C++ que obtém o hostname da máquina e imprime o nome da máquina em que executou.

Questão 02:

1. Tamanho aleatório do array: deve ser de pelo menos 6.500 elementos

1. A escolha do número: seu array é de tamanho n. O número escolhido pelo processo de rank 0 deve ser o elemento da posição (n/2) - 40

Questão 03:

O tamanho inicial do seu grid deve ser de 70 x 70. (versão serial)
$S_{survive}$ = {5,6,7}
S_{birth} = {2,3,4}

Número de gerações para executar: 300

Na implementação com OpenMP:
Você deve simular tamanhos de grids (nxn) = { 64, 96, 128} e número de threads = { 3 , 4}

TOPOLOGIA DE RESOLUÇÃO DAS QUESTÕES (via SLURM)
Nas questões de OpenMP seu programa deve sempre executar obrigatoriamente no nó compute01
Nas questões de MPI, a topologia de toda a sua resolução será 6 processos.
A tag MPI na questão 02 deve ser 70. Atenção: você pode criar outras TAGs para resolver o problema. Basta que uma delas seja esse valor.

As questões 02 e 03 devem ser submetidas sempre via SLURM. Seu job obrigatoriamente deve receber um nome da seguinte forma:
PROVA_D_<PRIMEIRO_NOME><ULTIMO_NOME> fazendo a substituição das tags por seus primeiro e último nomes, respectivamente.
Todos os Jobs devem, obrigatoriamente, executar na partição PROVA. Caso seu job não seja executado nessa partição, haverá decréscimo de 0,5 ponto por questão.

Resoluções incompletas serão consideradas para fins de correção, mas podem no máximo valor 50% da questão em si.
A resolução completa, porém sem submissão no cluster, também serão corrigidas e pontuadas em até 50% da questão.

Atenção aos itens que devem ser personalizados:

E

Questão 01:

Se você for o administrador, siga as instruções do slide. Caso contrário:

Prepare um job para executar na fila PROVA assim que ela estiver disponível. O seu job deve executar na topologia MPI descrita na caixa "Topologia" (abaixo), e deve ser um programa C++ que obtém o hostname da máquina e imprime o nome da máquina em que executou.

Questão 02:

1. Tamanho aleatório do array: deve ser de pelo menos 10.000 elementos

1. A escolha do número: seu array é de tamanho n. O número escolhido pelo processo de rank 0 deve ser o elemento da posição n/2

Questão 03:

O tamanho inicial do seu grid deve ser de 80 x 80. (versão serial)
$S_{survive}$ = {4, 5}
S_{birth} = {3, 4}

Número de gerações para executar: 170

Na implementação com OpenMP:
Você deve simular tamanhos de grids (nxn) = {96, 128, 256} e número de threads = { 2, 3 , 4}

TOPOLOGIA DE RESOLUÇÃO DAS QUESTÕES (via SLURM)
Nas questões de OpenMP seu programa deve sempre executar com 4 threads no nó compute01
Nas questões de MPI, a topologia de toda a sua resolução será 4 processos, dois processos por nó.
A tag MPI na questão 02 deve ser 120. Atenção: você pode criar outras TAGs para resolver o problema. Basta que uma delas seja esse valor.

As questões 02 e 03 devem ser submetidas sempre via SLURM. Seu job obrigatoriamente deve receber um nome da seguinte forma:
PROVA_E_<PRIMEIRO_NOME><ULTIMO_NOME> fazendo a substituição das tags por seus primeiro e último nomes, respectivamente.
Todos os Jobs devem, obrigatoriamente, executar na partição PROVA. Caso seu job não seja executado nessa partição, haverá decréscimo de 0,5 ponto por questão.

Resoluções incompletas serão consideradas para fins de correção, mas podem no máximo valor 50% da questão em si.
A resolução completa, porém sem submissão no cluster, também serão corrigidas e pontuadas em até 50% da questão.