Supercomputação – Avaliação Final

Instruções

Bem-vindo(a)s à prova final da disciplina de Supercomputação. Leia as instruções abaixo:

SOBRE HORÁRIOS:

- Abertura das salas (07h15); início da prova (07h30); final (09h30); limite (10h30);
- O tempo máximo deve ser utilizado para fazer a prova, preparar a submissão e entregá-la;

SOBRE SUBMISSÕES DA PROVA:

- A submissão da prova deve ser feita impreterivelmente até às 10h30 de 24/maio/2024 (horário de Brasília). NÃO serão aceitas submissões após este horário;
- O aluno pode fazer múltiplas submissões. O sistema considerará a última como oficial;
- A submissão da prova deve ser um ZIP de uma pasta principal**, nomeada com seu nome completo**, e organizada com uma subpasta para cada questão. Na pasta da questão deve ter (i) o código-fonte, (ii) o arquivo de submissão ao Slurm (.slurm) (se aplicável), (iii) o arquivo de saída do Slurm (ou um print comprovando a execução) e (iv) um TXT com suas respostas textuais (se aplicável). ATENÇÃO: nas questões teóricas, pode enviar apenas o TXT das respostas na pasta; na questão de GPU envie seu *jupyter notebook* baixado do Colab.

SOBRE A RESOLUÇÃO DA PROVA:

- É permitida a consulta ao material da disciplina (tudo o que estiver no repositório do Github da disciplina e no site http://insper.github.io/supercomp). Isso também inclui suas próprias soluções aos exercícios de sala de aula, anotações e documentações de C++ e bibliotecas pertinentes em sites oficiais;
- Execuções de código no cluster necessitam, obrigatoriamente, serem realizadas via submissão de Jobs não interativos. A execução de códigos da prova diretamente na linha de comando do SMS-HOST fere a boa prática de utilização. O log do cluster será monitorado, e caso seja constatado que o aluno rodou código sem submeter o job, a nota máxima da questão será 50% do valor original. Ex: a questão vale 3, e o aluno acertou tudo, mas executou de forma incorreta diretamente no nó principal, então a nota da questão será 1,5.

SOBRE QUESTÕES DE ÉTICA E PLÁGIO:

- A prova é individual. Qualquer consulta a outras pessoas durante a prova constitui violação do código de ética do Insper;
- Qualquer tentativa de fraude, como trechos idênticos ou muito similares, ou uso de GenAI, implicará em NOTA ZERO na prova a todos os envolvidos, sem prejuízo de outras sanções.

BOA PROVA! \o/

Questões

A interpretação do enunciado faz parte da avaliação. Em caso de dúvida, pode assumir premissas e explicá-las nos comentários.

Questão 1 – Submissão de Jobs (Total: 1 ponto)

Esta questão dá dicas para as próximas quando você precisará submeter jobs no cluster. Além disso, avalia sua compreensão de como solicitar recursos ao cluster via arquivo ".slurm".

Considere que:

- O parâmetro "`--mem=<tamanho>[unidades]`" especifica a memória real necessária por nó. As unidades padrão são megabytes. Diferentes unidades podem ser especificadas usando o sufixo [K|M|G|T].
- Você pode definir o número de threads usando a variável de ambiente OMP_NUM_THREADS. Uma forma simples é antes de chamar o executável fazer o "export" da variável de ambiente da seguinte forma "`export OMP_NUM_THREADS=<number of threads to use>`".

Pede-se:

- Crie um arquivo "script.slurm" que solicita 2 máquinas, com 5 cores por máquina, e uma quantidade de 3 gigas de memória. O job deve ser executado numa partição chamada "prova", exportando a variável de ambiente OMP_NUM_THREADS para usar 10 threads, e solicitar a execução de um código fictício denominado "./executavel"
- Nesta questão entregue apenas o .slurm criado
- NOTA: não é necessário submeter o job no cluster! Esta é uma questão teórica! ;D

Questão 2 – Paralelização Híbrida com MPI e OpenMP (Total: 3 pontos)

Desenvolva um programa que utiliza MPI para distribuir partes de um grande vetor (tamanho=100000) entre múltiplos processos e usa OpenMP dentro de cada processo para calcular a soma do quadrado de cada elemento do subvetor recebido. Cada processo deve enviar seu resultado parcial de volta ao processo raiz, que calculará a soma total dos quadrados.

Configuração do job: o código deve usar 4 processos, e 5 cores por máquina. A escolha do número máximo de threads está aberta. Considere que o nó principal também realizará parte dos cálculos.

Questão 3 – Paralelização de código sequencial (Total: 2.5 pontos)

Contexto:

Você recebeu um código que executa a multiplicação de matrizes de forma sequencial. Este código é usado em um aplicativo de processamento de dados para realizar transformações lineares em grandes conjuntos de dados. Devido ao tamanho das matrizes envolvidas, a execução atual é ineficiente e demorada.

Objetivo:

Seu trabalho é paralelizar este código usando OpenMP, aplicando estratégias sofisticadas de pragmas para otimizar a execução. Você deve identificar os melhores pontos para introduzir paralelismo e escolher os pragmas mais adequados para maximizar a eficiência do código.

Configuração do job: seu código deve ser testado sob 3 tamanhos de matrizes diferentes (escolha quais você quiser) ***E*** 3 configurações distintas de ambientes, sempre submetendo-os na partição "prova" e alocando uma máquina, mas usando 2 cores, 4 cores ou 8 cores.

NOTA: espera-se, portanto, 9 execuções!

Código sequencial: obtenha neste link → https://colab.research.google.com/drive/1jxLtNYW0PaoCWBUEMRw2j4FVtiLiR3rC?usp=sharing

Análise do resultado: apresente também num arquivo TXT a comparação dos tempos de execução sequencial, e nas 9 execuções. O tempo de execução agiu conforme você esperava? Comente e justifique potenciais distorções de expectativa x realidade.

Questão 4 – Normalização de Vetor com Thrust (Total: 2 pontos)

1. Acesse o link abaixo e faça uma cópia no seu Google Drive: https://colab.research.google.com/drive/14_EZNglXn2VXe3kpDW3XgEsRkB6R1jjp?usp=sharing
2. Complete o código seguindo a especificação. ATENÇÃO: você provavelmente precisará complementar os imports para o código rodar!
3. Baixe sua cópia do notebook preenchido e executado, e disponibilize na pasta desta questão

Questão 5 – Fundamentos de Paralelismo (Total: 1.5 pontos)

- **Parte A (0,75 ponto)**: Descreva o modelo de memória compartilhada e memória distribuída, ressaltando seus prós e contras. Qual modelo cada uma das seguintes bibliotecas usa: OpenMP, MPI e Thrust?
- **Parte B (0,75 ponto)**: Explique o que é escalonamento dinâmico em OpenMP e como ele pode ser vantajoso em aplicações de processamento de dados com variabilidade de carga entre as iterações.