Universidade Federal do ABC
MCZA020-13 - Programação Paralela
2019.Q1

Lista de Exercícios 3

Prof. Emílio Francesquini

11 de março de 2019

Lista de termos cuja definição você **deve** saber:

- Determinismo/Não determinismo
- MPI
- SPMD
- Comunicador, rank, tag, buffer
- Comunicação assíncrona e síncrona
- Comunicação coletiva e ponto-a-ponto
- Particionamento por blocos, cíclico ou bloco-cíclico
- Speedup e eficiência
- Overhead da versão paralela de um algoritmo
- *Unsafe* MPI Programs
- Impasses (*deadlocks*)

Exercícios

1. Altere o programa de cálculo de áreas baseado em trapézios visto em aula para que ele trate corretamente o caso em que `comm_sz` não divide `n`.
2. Modifique o programa visto em aula no qual cada processo imprimia o seu rank para que todos os ranks sejam impressos em ordem crescente.

3. [PP] Suponha que `comm_sz = 4` e suponha que `x` seja um vetor com $n = 14$ elementos.

   (a) Como os elementos de `x` seriam distribuídos entre os processos em um programa que utilizasse distribuição em blocos?

   (b) E no caso de distribuição cíclica?

   (c) Considerando o caso de uma distribuição bloco-cíclica bom tamanho de bloco 2, como ficaria a distribuição?

4. O que acontece caso o comunicador tenha apenas um processo e sejam chamadas funções de comunicação coletiva? O comportamento é o mesmo para todas as funções?

5. [PP] Suponha que `comm_sz = 8` e a quantidade de elementos é `n = 16`.

   (a) Desenhe um diagrama que explique como `MPI_Scatter` pode ser implementado usando comunicações baseadas em árvores. Você pode supor que a origem do scatter é o processo com rank 0.

   (b) Faça o mesmo para o `MPI_Gather`, neste caso com o processo 0 como destino.

6. Utilizando apenas as operações send e receive do MPI, descreva como fazer a implementação das operações de comunicação coletiva broadcast, scatter, gather e reduce.

7. Considere as topologias de interconexão baseadas em anel, hipercubo e grafo completo. Descreva uma forma eficiente para realizar, levando em consideração a topologia dessas redes, as operações de scatter, gather e broadcast.

8. Utilizando MPI, implemente um algoritmo que determine o número de vezes que um determinado inteiro ocorre em um vetor A com n elementos. Você pode assumir que o seu algoritmo começa com os elementos já distribuídos entre os p processos ($\frac{n}{p}$ em cada).

9. Desenvolva um algoritmo em MPI utilizando apenas `send` e `receive` que encontre o maior e o menor valor de um conjunto S com n elementos com p processadores. Você pode assumir que o seu algoritmo começa com os elementos já distribuídos entre os p processos ($\frac{n}{p}$ em cada). Analise a complexidade do algoritmo proposto em termos do número de comunicações e complexidade local.

10. Desenvolva um algortimo em MPI, utilizando p processadores para calcular $n!$.

11. Desenvolva um algoritmo em MPI utilizando p processadores, para calcular o produto de uma matriz A, $n \times n$ de inteiros por um vetor B de n inteiros e obter um vetor C de n inteiros. Neste algoritmo, cada processador recebe $\frac{n}{p}$ linhas da matriz A e o vetor B inteiro, e obtém $\frac{n}{p}$ elementos do vetor C, onde:

$$C[i] = A[i,1] \times B[1] + A[i,2] \times B[2] + \ldots + A[i,n/p] \times B[n/p]$$

Analise a complexidade do algoritmo proposto.

12. [PP] Costuma-se dizer que o speedup ideal é alcançado quando ele é igual ao número de processadores. É possível fazer algo melhor?

    (a) Considere um programa paralelo que computa a soma de um vetor. Se medirmos apenas o tempo que leva para somar do vetor (ou seja, ignoramos o tempo que leva para ler o vetor e para escrever o resultado) como poderíamos fazer este programa alcançar um speedup superior a p, onde p é o número de processos?

    (b) Um programa que alcança speedup superior a p é possuidor de um speedup superlinear. O nosso exemplo de speedup superlinear acima apenas alcançou speedup por ter superado certas "limitações de recursos". Quais são essas limitações? É possível escrever um programa que tenha speedup superlinear sem que tais limitações sejam superadas?

13. [PP] O algoritmo de ordenação sequencial baseado em transposição par-ímpar de uma lista com n elementos pode ser capaz de ordenar a lista em bem menos de n fases. Como um exemplo extremo, se a lista estiver ordenada o algoritmo leva 0 fase.

    (a) Escreva uma função sequencial `is_sorted` que determina se uma lista está em ordem ou não.

    (b) Modifique o algoritmo sequencial baseado em tansposição par-ímpar para que ele verifique se a lista está ordenada após cada fase.

    (c) Caso este programa seja testado em diversas listas aleatórias com n elementos cada, qual é proporção (aproximada) de execuções que terá um aumento no desempenho?

14. Consulte a tabela de speedups e eficiências do algoritmo de ordenação paralela por transposição discutido em aula. O programa é capaz de obter speedups lineares? Ele é escalável? Ele é fortemente escalável? Ele é fracamente escalável?