Programação Paralela e Distribuída

Programação Paralela com MPI – Modelo Mestre Escravo

Exercício 1 – *Rank Sort* paralelo

O objetivo do trabalho é implementar, usando a biblioteca MPI, uma versão paralela, usando o modelo mestre escravo, do algoritmo de ordenação *Rank Sort*. Após implementado, o programa deve ser executado no cluster para realização das medições de desempenho e, por fim, deve-se gerar o gráfico de *speed up*.

O algoritmo de ordenação *Rank Sort* funciona da seguinte forma: O número de valores que são menores que um valor selecionado da lista é contado. Esta contagem provê a posição do número selecionado na lista ordenada. Primeiro, o valor de a[0] é lido e comparado com cada um dos outros números, a[1] ... a[n-1], armazenando o número de valores menores que a[0]. Suponha que este número é x. Este é o índice da localização na lista ordenada final. O número a[0] é copiado na lista ordenada final b[0] ... b[n-1], na localização b[x]. Ações são repetidas com todos os outros valores da lista de entrada. Considere um lista sem repetições de números.

O algoritmo sequencial é descrito abaixo:

```
for(i=0; i<n; i++)
{
  x=0;
  for(j=0; j<n; j++)
     if(a[i]>a[j])
      x++;
  b[x] = a[i];
}
```

O programa deverá receber como parâmetros de entrada um número N, representando o número de valores a ser testado e o nome de um arquivo, que contém a lista de valores inteiros a serem ordenados. Utilizar 3 casos de teste para realização das medições no cluster. A saída que deve ser gerada é a **lista ordenada (crescente) dos valores de entrada** (1 valor por linha) e também o **tempo de execução da aplicação**.

Deverá ser implementada uma versão paralela para o problema da seguinte forma: cada processo escravo recebe um conjunto contendo uma fração do número total de tarefas do processo mestre, executa todas as tarefas e retorna o resultado para o processo mestre. Observação: todos os processos recebem número de tarefas similar (depende da velocidade dos escravos - cada escravo recebe uma tarefa por vez). O número de tarefas é definido por N / 4*M, onde N é o número de elementos do vetor de entrada e M é o número de processadores usados na execução.

Os itens para avaliação são:

- implementação da versão sequencial do algoritmo;
- implementação da versão paralela do algoritmo
- medição dos tempos de execução para as versões sequencial e paralelas (usando 2, 4 e 8 processadores) para os 3 casos de teste (arquivos de entrada);
- cálculo do *speed up* para os casos de teste e número de processadores;
- geração dos gráficos de *speed up* e eficiência;
- otimização do código;
- clareza do código (utilização de comentários e nomes de variáveis adequadas);
- documento descrevendo o problema e a solução.

PS: A nota do Trabalho 2 será uma composição das notas dos exercícios que serão realizados em aula. **O exercício pode ser feito em duplas.**