Facultad de Ciencias

Computación Distribuida

PRACTICA 04

Semestre 2024 – 1

Profesor:
Luis Germán Pérez Hernández

Ayudantes:
Daniel Michel Tavera
Yael Antonio Calzada Martín

Alumnos

Marco Silva Huerta
316205326

Edgar Montiel Ledesma
317317794

Carlos Cortés
420004846

30 de Noviembre de 2023

Ordenamiento distribuido

Descripción de la Práctica

El equipo deberá implementar una versión distribuida del algoritmo de ordenamiento por mezcla (Merge Sort), como sigue:

- Se genera de forma aleatoria un arreglo de números enteros a ordenar.
- Se imprime en pantalla el arreglo original (antes de ordenar)
- Se reparte el arreglo entre los nodos de la forma más equitativa posible.
- Cada nodo utiliza algún algoritmo de ordenamiento secuencial para ordenar sub-arreglo local.
- Se utiliza el procedimiento de mezcla para ir incorporando los resultados parciales.
- Se imprime el resultado final, que debe ser el arreglo original pero ordenado.

Ejecución del programa

Forma de compilar

```
mpicc Practica04_EdgarMontiel_CarlosCortes_MarcoSilva.c -o Merge
```

Forma de ejecutar

```
./Merge
```

Funcionamiento

1. Se incluyen las bibliotecas necesarias: stdio.h para entrada y salida estándar, stdlib.h para funciones relacionadas con la memoria dinámica, y mpi.h para la programación paralela con MPI (Message Passing Interface).
2. Se define una función print_array para imprimir un arreglo de enteros.
3. Se define una función merge que realiza la mezcla de dos arreglos ordenados en un solo arreglo ordenado. Es parte del algoritmo de ordenación por mezcla.
4. Se define una función merge_sort que implementa el algoritmo de ordenación por mezcla de manera recursiva. Divide el arreglo en dos mitades, ordena cada mitad por separado y luego combina las dos mitades ordenadas usando la función merge.
5. Dento del main se inicia MPI, se obtiene el rango del proceso actual (rank) y el número total de procesos (size).
   a) El proceso con rango 0 (nodo maestro) genera un arreglo de números aleatorios si es el proceso con rango 0 y luego imprime el arreglo original.

b) Se calcula el tamaño de los subarreglos locales (local_size) y se crea un arreglo local (local_array). Luego, el arreglo original se divide entre los nodos utilizando MPI_Scatter.

c) Cada nodo ordena su subarreglo local utilizando la función merge_sort.

d) Los resultados parciales (los subarreglos ordenados localmente) se recopilan de nuevo en el nodo maestro utilizando MPI_Gather.

e) El nodo maestro imprime el arreglo ordenado final si es el proceso con rango 0. Luego, se finaliza MPI y se devuelve 0 para indicar la finalización exitosa del programa.

Pseudocódigo del Algoritmo

```
1. Generar un arreglo de números enteros aleatorios.
    1.1. Definir el tamaño del arreglo.
    1.2. Para cada posición en el arreglo, generar un número entero aleatorio
         y asignarlo a esa posición.

2. Imprimir el arreglo original.
    2.1. Recorrer el arreglo y imprimir cada elemento.

3. Dividir el arreglo entre los nodos de la forma más equitativa posible.
    3.1. Determinar el número de nodos disponibles.
    3.2. Calcular el tamaño de los sub-arreglos dividiendo el tamaño del
         arreglo original entre el número de nodos.
    3.3. Para cada nodo, asignarle un sub-arreglo del arreglo original.

4. Para cada nodo:
    4.1. Ordenar el sub-arreglo local utilizando un algoritmo de ordenamiento
         secuencial.
        4.1.1. Puede ser cualquier algoritmo de ordenamiento secuencial, como el
               ordenamiento por inserción, burbuja, etc.

5. Utilizar el procedimiento de mezcla para incorporar los resultados parciales.
    5.1. Mientras haya más de un sub-arreglo:
        5.1.1. Tomar dos sub-arreglos.
        5.1.2. Mezclarlos en un nuevo sub-arreglo ordenado.
        5.1.3. Reemplazar los dos sub-arreglos originales con el nuevo sub-arreglo
               en la lista de sub-arreglos.

6. Imprimir el resultado final.
    6.1. Recorrer el arreglo final y imprimir cada elemento.
```