Universidad Nacional Autónoma de México

Facultad de Ciencias

Computación Distribuida

Reporte de Práctica 4

1. Equipo

Integrantes del equipo:

- Bonilla Reyes Dafne - 319089660
- García Ponce José Camilo - 319210536
- Ortega Venegas Rodrigo Aldair - 318036104
- Rivera Mora Jesús Alberto - 313208641

2. Desarrollo

2.1 Descripción del desarrollo de la práctica y su funcionamiento:

a) *Funciones auxiliares:*

1) La función `merge` toma dos arreglos como parametros `a1[],a2[]` y los fusiona en un nuevo arreglo `merged`, los índices `x` e `y` representan las posiciones actuales en `a1` y `a2`, mientras que `z` representa la posición en el arreglo `merged`.
2) La función `mergeSort` implementa el algoritmo **Merge Sort** de manera recursiva. EL algoritmo divide el arreglo en dos mitades, ordena cada mitad por separado y luego fusiona las mitades ordenadas.

b) *Configuración:*

1) Se incluyen las bibliotecas necesarias (`stdio.h, stdlib.h, mpi.h, time.h, string.h`).
2) Se definen constantes `TAG_IND_INI, TAG_IND_FIN, TAG_FRAC_ORG,` y `TAG_FRAC_ORD` para etiquetas de mensajes MPI.

c) *Inicialización:*

1) Inicializa MPI con `MPI_Init` y se obtiene el tamaño del comunicador y el rango del proceso.
2) Verifica la entrada del usuario para asegurarse de que se proporcionen el número correcto de argumentos de línea de comandos.
3) Conviertimos los argumentos de línea de comandos a enteros (n y i).
4) Verifica que el número de nodos (n) sea igual al número de procesos MPI.
5) Verifica que el número de nodos sea mayor a 1 y que el tamaño del arreglo sea mayor al número de nodos.

d) *Generación de arreglo a ordenar:*

1) Se declara un arreglo `int original [i];` de longitud `i`
2) Asignamos a cada posición del arreglo `orginal[]` un número alatorio entre 0 y 99 y se muestra este arreglo en la terminal.

e) *Distribución del arreglo:*

1) Creamos un arreglo de arreglos `int fracciones[size][2]` auxiliar para guardar los índices del arreglo original que le corresponden a cada nodo.
2) El nodo maestro (nodo 0) calcula la fracción del arreglo original que le corresponde ordenar a cada nodo y almacena los índices que le corresponden a cada nodo en `fracciones`.
3) El nodo 0 envia a cada nodo de que indice a que indice le corresponde ordenar, declaramos las variables `int inicio = [j][0] int fin =[j][1]`, donde todos los nodos incluido el nodo 0 reciben sus rangos.

f) *Ordenamieto:*

1) Cada nodo calcula el tamaño de la fraccion del arreglo que le corresponde ordenar
2) El nodo 0 distribuye la fraccion del arreglo que le corresponde a cada nodo ordenar y se asigna su parte a ordenar. Los nodos que son diferentes que el nodo 0 reciben su fracción del arreglo.
3) Cada nodo ordena su fraccion del arreglo usando la función `mergeSort`.
4) Cada nodo envia su fraccion del arreglo ordenada al nodo 0.
5) El nodo 0 incluye su fracción ordenada en `*ordenado`, recibe las fracciones ordenadas de cada nodo y las va uniendo con la función `merge` a el arreglo `*ordenado`.

g) *Resultados:*

1) Se imprime el arreglo generado con números aleatorios ordenado.

h) *Finalización:*

1) El programa MPI finaliza utilizando `MPI_Finalize()`.

2.2 Compilación y salidas:

- Compilar desde `/Practica4`:

```
root@unam:~/Practica4$
                    mpicc Practica4_DafneBonilla_CamiloGarcia_RodrigoOrtega_JesusRivera.c
```

- Para ejecutar desde `/Practica4`:

```
root@unam:~/Practica4$                mpirun -np n --oversubscribe ./a.out n i
```

Donde `n` es el número de nodos e `i` el tamaño del arreglo.

2.3 Ejemplos de salidas:

- Ejemplo con 10 nodos y un arreglo de tamaño 25:

```
root@unam:~/Practica4$                mpirun -np 10 --oversubscribe ./a.out 10 25
               Arreglo original: 80 98 32 66 28 90 61 29 91 15 50 93 73 79 52 13 33 93 68
    83 49 61 4 36 81
               Arreglo ordenado: 4 13 15 28 29 32 33 36 49 50 52 61 61 66 68 73 79 80 81
    83 90 91 93 93 98
```

- Ejemplo con 5 nodos y un arreglo de tamaño 50:

```
root@unam:~/Practica4$                mpirun -np 5 --oversubscribe ./a.out 5 50

               Arreglo original: 47 27 85 43 66 82 90 25 11 39 46 3 8 80 98 9 11 44 90 26
    44 1 22 56 73 20 66 7 62 36 26 9 63 63 52 81 97 43 59 9 82 57 64 42 37 62 52 48 59 94
               Arreglo ordenado: 1 3 7 8 9 9 9 11 11 20 22 25 26 26 27 36 37 39 42 43 43
    44 44 46 47 48 52 52 56 57 59 59 62 62 63 63 64 66 66 73 80 81 82 82 85 90 90 94 97 98
```

2.4 Comentarios:

Usamos la estrategia de **estrella** para repartir el arreglo entre los nodos.