Arquitectura de Computadores

Grado en Ingeniería Informática

GRUPO A

Práctica 4: Resolución de problemas mediante el modelo de comunicación colectivo

UNIVERSIDAD NEBRIJA

Tabla de contenido

I. Introducción

La presente práctica profundiza en el modelo de comunicación colectivo presentado en la práctica anterior. Se trabajará con las funciones MPI `MPI_Allgather` y `MPI_Alltoall`, con las que se pueden realizar otro tipo de comunicaciones entre procesos.

II. Objetivos

- Compilación de programas MPI.
- Ejecución de programas en varios procesos de forma paralela.
- Estructura de un programa MPI.
- Estudio de las funciones aportadas en este documento.

III. Entorno

Previo a la implementación de los ejercicios propuestos, es necesario tener instalado un compilador de C++ como G++, y el paquete MPICH en un entorno Unix, ya sea a través de una máquina virtual o una partición en los equipos personales.

```
sudo apt update
sudo apt install g++
sudo apt install mpich
mpicxx --version //comprueba que la instalación es correcta
```

A. Compilación de programas

Compilación de un programa con MPICH:

```
mpicxx nombre_del_archivo.cpp -o nombre_del_ejecutable
```

B. Ejecución de programas

Ejecutar el archivo compilado generado:

```
mpirun -np numero_procesos_en_paralelo ./nombre_del_ejecutable
```

IV. Implementación

A. Funciones MPI básicas

Entre las funciones básicas que se pueden encontrar en la librería MPI podemos encontrar las siguientes:

Función	Descripción
MPI_Init	Inicializa el entorno MPI
MPI_Comm_rank	Obtiene el rango del proceso actual en un comunicador
MPI_Comm_size	Obtiene el número total de procesos en un comunicador
MPI_Finalize	Finaliza el entorno MPI
MPI_Allgather	Recopila datos de todos los procesos y los distribuye a todos
MPI_Alltoall	Cada proceso envía distintos elementos al resto

La interfaz de las funciones se puede ver a continuación:

MPI_Init

```
#include <mpi.h>

MPI_Init(&argc, &argv);
```

- `argc`: puntero al número de argumentos de línea de comandos.
- `argv`: puntero a un array de punteros a cadenas de caracteres.

MPI_Comm_rank

```
#include <mpi.h>

MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

- `MPI_COMM_WORLD`: comunicador en el que se desea obtener el rango.
- `rank`: puntero al entero donde se almacenará el rango del proceso.

MPI_Comm_size

```
#include <mpi.h>

MPI_Comm_size(MPI_COMM_WORLD, &size);
```

- `MPI_COMM_WORLD`: comunicador en el que se desea obtener el tamaño.
- `size`: puntero al entero donde se almacenará el tamaño del comunicador.

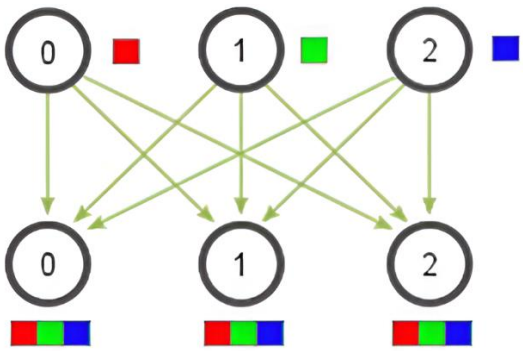
MPI_Finalize

```
#include <mpi.h>

MPI_Finalize();
```

MPI_Allgather

Recopila datos de todos los procesos y los distribuye a todos los procesos.



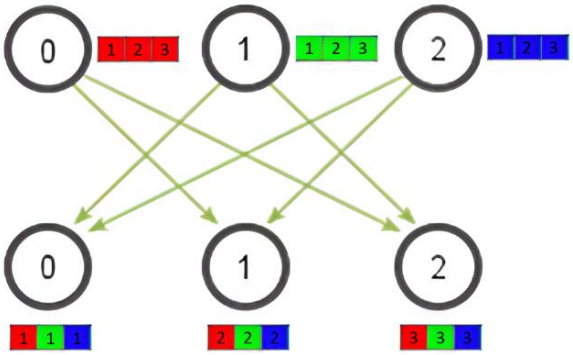

```
#include <mpi.h>

MPI_Allgather(&sendbuf, sendcount, sendtype, &recvbuf, recvcount,
recvtype, MPI_COMM_WORLD);
```

- `sendbuf`: puntero al buffer donde se almacenarán los datos a enviar.
- `sendcount`: número de elementos que se enviarán desde cada proceso.
- `sendtype`: tipo de dato de los elementos en el buffer a enviar. Algunos ejemplos son `MPI_INT`, `MPI_FLOAT`, `MPI_DOUBLE`, `MPI_CHAR`, `MPI_BYTE`.
- `recvbuf`: puntero al buffer donde se almacenarán los datos recibidos.
- `recvcount`: número de elementos que recibirá cada proceso.
- `recvtype`: tipo de dato de los elementos en el buffer de recepción. Algunos ejemplos son `MPI_INT`, `MPI_FLOAT`, `MPI_DOUBLE`, `MPI_CHAR`, `MPI_BYTE`.
- `MPI_COMM_WORLD`: comunicador utilizado para la comunicación.

MPI_Alltoall

Distribuye datos diferentes cada proceso al resto.




```
#include <mpi.h>

MPI_Alltoall(&sendbuf, sendcount, sendtype, &recvbuf, recvcount,
recvtype, MPI_COMM_WORLD);
```

- `sendbuf`: puntero al buffer donde se almacenarán los datos a enviar.
- `sendcount`: número de elementos que se enviarán desde cada proceso.
- `sendtype`: tipo de dato de los elementos en el buffer a enviar. Algunos ejemplos son `MPI_INT`, `MPI_FLOAT`, `MPI_DOUBLE`, `MPI_CHAR`, `MPI_BYTE`.
- `recvbuf`: puntero al buffer donde se almacenarán los datos recibidos.
- `recvcount`: número de elementos que recibirá el proceso raíz.
- `recvtype`: tipo de dato de los elementos en el buffer de recepción. Algunos ejemplos son `MPI_INT`, `MPI_FLOAT`, `MPI_DOUBLE`, `MPI_CHAR`, `MPI_BYTE`.
- `root`: rango del proceso que recibirá los datos recopilados.
- `MPI_COMM_WORLD`: comunicador utilizado para la comunicación.

V. Entregables

Ejercicio 1 (2 puntos)

Desarrollar un programa donde existan 4 procesos y cada uno de ellos cuente con un número que solo conoce él. Tras esto, se deberá aplicar una de las funciones de comunicación entre procesos vista, para lograr que cada proceso cuente con todos los números. Finalmente, cada proceso imprimirá por pantalla los valores de su buffer de recepción.

Ejercicio 2 (3 puntos)

Desarrollar un programa que realice la transposición de la matriz inicial de la izquierda y resulte en la de la derecha. Realizar el proceso en paralelo, distribuyendo la matriz entre 4 procesos.

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{pmatrix}$$

Debido a la concurrencia del programa, para mostrar los resultados claros de cara a la memoria, compila y ejecuta varias veces el programa añadiendo antes del `MPI_Finalize` el siguiente código, y cambiando el rango de comprobación cada vez.

```
if (rango == 0) {
      for (int i=0; i < 4; i++) {
            std::cout << recv_data[i] << " ";
      }
      std::cout << "\n";
}
```

Ejercicio 3 (5 puntos)

Desarrollar un programa de conversión paralela de número binario a decimal. Se debe hacer uso de las funciones mostradas en este documento y cumplir los siguientes requisitos:

- El número de procesos con el que se lance la aplicación será igual al número de bits del número binario. Ej: para el número binario 10 son necesarios 2 procesos.
- Cada proceso tendrá un array que contiene el número en binario a convertir en decimal.
- Cada proceso estará “especializado” en realizar la operación 2 elevado al rango. Por lo que se deberá repartir los bits de tal forma que este funcionamiento sea útil para la conversión.
- Tras realizar la operación se devolverán los nuevos valores al proceso de donde proceden.
- Por último, cada proceso sumará los datos del array, y mostrará por pantalla el número en decimal.
- El ejercicio se debe resolver con los números binarios: 1010, 1100, 0001, y 1111.

VI. Criterios de evaluación

A. Calificación mínima

El alumno se le considerará como aprobado en las prácticas de la asignatura si la media de las cinco prácticas es igual o superior a 5. Si se obtiene una nota media menor, este deberá presentarse a la convocatoria extraordinaria independientemente de la nota obtenida en la parte teórica de la asignatura.

En convocatoria extraordinaria la nota media se realizará con las prácticas ya aprobadas, y las nuevas entregadas. De igual forma, la nota media de estas deberá ser igual o superior a 5 para considerarse aprobadas.

En el caso que un alumno suspenda en convocatoria extraordinaria la parte teórica de la asignatura y apruebe la parte práctica, el alumno deberá presentarse a la siguiente convocatoria con ambas partes. No se guarda la nota de las prácticas de un curso para otro.

B. Asistencia

La asistencia a las prácticas es obligatoria, en caso de falta justificada la recuperación de esta se realizará en una fecha propuesta por el profesor. En el caso de no estar justificada la asistencia del alumno, obtendrá un 0 en dicha práctica.

C. Advertencia sobre plagio

La Universidad Antonio de Nebrija no tolerará en ningún caso el plagio o copia. Se considerará plagio la reproducción de párrafos a partir de textos de auditoría distinta a la del estudiante (internet, libros, artículos…), cuando no se cite la fuente original de la que provienen. El uso de las citas no puede ser indiscriminado. Si se detecta que el trabajo ha sido copiado de un compañero, ambos obtendrán una calificación de 0 en la práctica e irán a directamente a convocatoria extraordinaria.

En caso de detectarse este tipo de prácticas, se considerará Falta Grave y se podrá aplicar la sanción prevista en el Reglamento del Alumno.

D. Ponderación

En el caso de que haya ejercicios correctamente desarrollados y no se justifique su implementación se calificará sobre la mitad de la puntuación de ese ejercicio.

Si la solución del ejercicio no es correcta por pequeños detalles de implementación y su justificación es correcta. La calificación se aplicará según criterio del profesor.

Si el ejercicio solo cuenta con una justificación teórica y no con una implementación práctica, el ejercicio obtendrá 0 puntos.

E. Entregas Ordinarias

Las prácticas tendrán como fecha máxima de entrega 24 horas después de la finalización de la clase. Los alumnos deberán subir la práctica en formato ZIP al campus virtual.

El ZIP de entrega debe contener:

- Memoria de la práctica en formato PDF (índice, enunciados, resultados, conclusiones, etc).
- Código de cada ejercicio.

No se admitirán prácticas fuera de fecha y horario establecido.

F. Entregas Extraordinarias

Las prácticas suspensas serán entregadas en el período de recuperación (fecha por determinar). Los alumnos deberán subir las prácticas en formato ZIP al campus virtual.

El ZIP de entrega debe contener:

- Memoria de la práctica en formato PDF (índice, enunciados, resultados, conclusiones, etc).
- Código de cada ejercicio.

No se admitirán prácticas fuera de fecha y horario establecido.