Hoja de Trabajo 2: OpenMPI y Computación Distribuida

Competencias a desarrollar:

Implementa y diseña programas distribuidos mediante intercambio de mensajes y memoria distribuida usando OpenMPI.

1. Instrucciones

Resuelva los siguientes ejercicios, haciendo las modificaciones usando MPI para crear soluciones paralelas de un programa secuencial. Deje evidencia de todo su trabajo.

La configuración de su entorno es intrínsecamente individual, mas puede apoyarse de sus compañeros con entornos similares. **Cada alumno debe entregar su copia individual** de la hoja en formato PDF. No olvide también subir su código.

En todos los incisos debe dejar copia de lo siguiente en donde aplique:

- Función(es) principal(es) del código. La parte que contiene el cálculo o tarea más importante. No incluya encabezados ni operaciones de interacción con el usuario.
  - Bloque de código modificado
- Captura de pantalla de la salida (x1)
- Descripción de lo que se observa
- Cualquier otro elemento o discusión que consideren adecuado.

2. Verificación e instalación de MPI

Linux/WSL:

Alguna versión de MPI viene instalada por defecto en muchas de las distribuciones de Linux. Verifique si ya tiene instalado MPI en su sistema mediante:

```
$ mpicc -v
```

Si no tiene instalado MPI en Linux o WSL, instálelo mediante:

```
$ sudo apt-get install openmpi*

$ sudo apt install openmpi-bin

$ sudo apt install mpich
```

Si les da error en la instalación, es por la ausencia de los archivos para compilar Fortran. Pueden resolver esto actualizando los paquetes luego de la falla:

```
$ sudo apt-get update
```

MacOS:

- Tener instalado Xcode y Xcode cmd tools

```
$ xcode-select --install
```

- Tener instalado el compilador GNU o el compilador Intel
- Descargar el paquete tar.gz desde el sitio de OpenMPI . **No olvide revisar que obtenga la versión más reciente.**
  - https://download.open-mpi.org/release/open-mpi/v4.1/openmpi-4.1.5.tar.gz
- Copiar a una ubicación temporal y desempacar el archivo

```
$ tar zxvf openmpi-4.1.5.tar.gz
```

- (**Este proceso es tardado**) Navegar al directorio creado y/o buscar el script de instalación. Especificar el compilador instalado y si se desea cambiar el directorio de instalación, modificar la bandera --prefix:

```
$ ./configure CC=gcc-11 CXX=g++-11 F77=gfortran
FC=gfortran --prefix=/usr/local
```

- (**Este proceso es tardado x2**) Si la creación de configuración es exitosa, compilar el código:

```
$ make all
```

- Crear y ejecutar la instalación:

```
$ sudo make install
```

- Verificar que el directorio de instalación esté en el PATH de la máquina:

```
$ echo $PATH
```

3. mpiHello.c

Todo programa de MPI usa siempre la misma estructura.

a. Descargue el programa mpiHello.c Este muestra la estructura básica de todo programa de MPI.

b. Compilación − para compilar, debemos usar MPICC, que es un wrapper que identifica el compilador que usaremos y las variables del entorno:

```
$ mpicc mpiHello.c -o mpiHello
```

c. Si no podemos o no queremos usar MPICC, podemos compilar directamente mediante GCC/G++:*

```
$ gcc mpiHello.c -o mpiHello –I /usr/bin/mpi –lmpi
```

d. *O bien, ejecutar el comando `mpicc -showme` para obtener un detalle de lo que el wrapper de mpicc hace “tras bambalinas”.

(+) En caso dado el archivo de biblioteca mpi.h no esté en ese directorio, pueden buscar en donde está usando el siguiente comando:

```
$ find / | grep mpi.h
```

e. Ejecución − la ejecución de programas MPI se hace con el siguiente comando. MPI utiliza un máximo de N procesos según el número de recursos físicos. Es preferible especificar siempre el número de procesos que ejecutarán un programa:

```
$ mpirun -np 4 mpiHello    //linux
$ mpirun -np 4 ./mpiHello  //WSL
```

4. Intercambio de mensajes

El paradigma usado por MPI para memoria distribuida es el de intercambio de mensajes. En MPI tenemos dos modelos de comunicación: a)Punto a Punto y b)Colectiva.

a. Modifique el programa mpiHello.c para que la parte (3) use el siguiente código:

```
if (rank == 0)
    {
    char mess[] = "Hello World";
    printf("%i sent %s\n", rank, mess);
    for (i = 1; i < num; i++)
    MPI_Send(mess, strlen(mess) + 1, MPI_CHAR, i, MESSTAG,
              MPI_COMM_WORLD);
    }
else
    {
```

```
        char mess[MAXLEN];
        MPI_Status status;
        MPI_Recv (mess, MAXLEN, MPI_CHAR, 0, MESSTAG,
MPI_COMM_WORLD, &status);
        printf ("%i received %s\n", rank, mess);
        }
```

b. Compile y ejecute el programa. Observe cómo ahora hay envío y recepción de mensajes basado en el ID de proceso (rank). Comente y discuta sus observaciones.

5. Mensajes con bloque y correspondencia

Para que un mensaje sea enviado y recibido correctamente, debemos asegurarnos que haya correspondencia entre el remitente (SEND) y el destinatario (RECV). Un proceso con recepción de mensaje se queda en espera de su correspondiente envío.

a. Modifique las siguientes líneas de código. Compile el programa y ejecútelo para observar lo que sucede:

```
if(rank == 0) -> if(rank %2 == 0)
(elimine o comente) for (i = 1; i < num; i++)
MPI_Send(mess, strlen(mess) + 1, MPI_CHAR, i, MESSTAG,
                MPI_COMM_WORLD); ->
-> MPI_Send(mess, strlen(mess) + 1, MPI_CHAR, rank+1,
                MESSTAG, MPI_COMM_WORLD);
```

b. El órden de los argumentos de envío es:
MPI_Send(mensaje, longitud mensaje, tipo de datos, destinatario, etiqueta, comunicador)
c. El órden de los argumentos de recepción es:
MPI_Recv(mensaje, longitud mensaje, tipo de datos, remitente, etiqueta, comunicador)
d. Observe como uno de los procesos se queda en espera y esto para por completo el programa. Haga los cambios necesarios para que los mensajes sean recibidos de forma correcta.

6. Entregar en Canvas

a. **PDF** con los incisos realizados y sus procedimientos, capturas, discusiones, etc..
b. Adicionalmente, tome una captura de pantalla de su sistema ejecutando el programa con correspondencia de mensajes. Modifique el programa para que imprima su nombre completo y carnet como parte del mensaje Hello World. Incluya la captura al final del PDF.
c. Incluya el **código fuente completo** en su entrega de Canvas. (.c/.cpp)