

Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
Programación paralela y distribuida
Ing. Sebastián Galindo

Corto # 3



Christopher García 20541
Semestre II

Octubre 2023

Primera Parte (30 puntos, 7.5pts c/u)

1. Describa con sus propias palabras, ¿qué es MPI? ¿Para qué lo utilizamos? ¿Cómo ayuda a la computación paralela? (No hay un mínimo o un máximo de oraciones/párrafos, pero si su respuesta es muy superficial se le descontaran puntos de esta pregunta)

- Es una interfaz de comunicación que funciona entre diferentes dispositivos y/o unidades de procesamiento a través de una red. Este se considera un middleware y con esto puede abstraer la lógica y el funcionamiento relacionada a la red donde se ejecuta. El principal uso es el intercambio de mensajes entre procesos. La computación paralela entra con la utilización del patrón divide and conquer, usualmente se da en Clusters y se aprovecha el procesamiento de múltiples dispositivos de la red y de memoria. Una de sus ventajas más grandes es que posee una alta dependencia de la red y esto trae consecuencias en la comunicación de mensajes.

2. Haga un diagrama o tabla en donde destaque las diferencias o similitudes entre OpenMP y Open MPI.

OpenMP	Open MPI
Equipos de threads	Colectivos de procesos/dispositivos
Scope (Espacio donde se ejecuta)	Grupos comunicadores
Directivas (#pragma)	Subrutinas (MPI_”x”)
Un solo equipo	Múltiples dispositivos

(Esta tabla proviene de mis apuntes de clase)

3. Detalle las definiciones para los siguientes conceptos

a. Communication Domain

- Existe una estructura de datos y se define qué procesos pueden comunicarse entre sí dentro del espacio. Se utiliza la directiva MPI_COMM_WORLD() para definir el espacio y todo lo que esté definido dentro de la red del espacio puede recibir comunicación. Si no se desea esto se incluye un Hosts File.

b. Rank

- Sirven como identificadores dentro de la red.

c. Subrutinas/Directivas

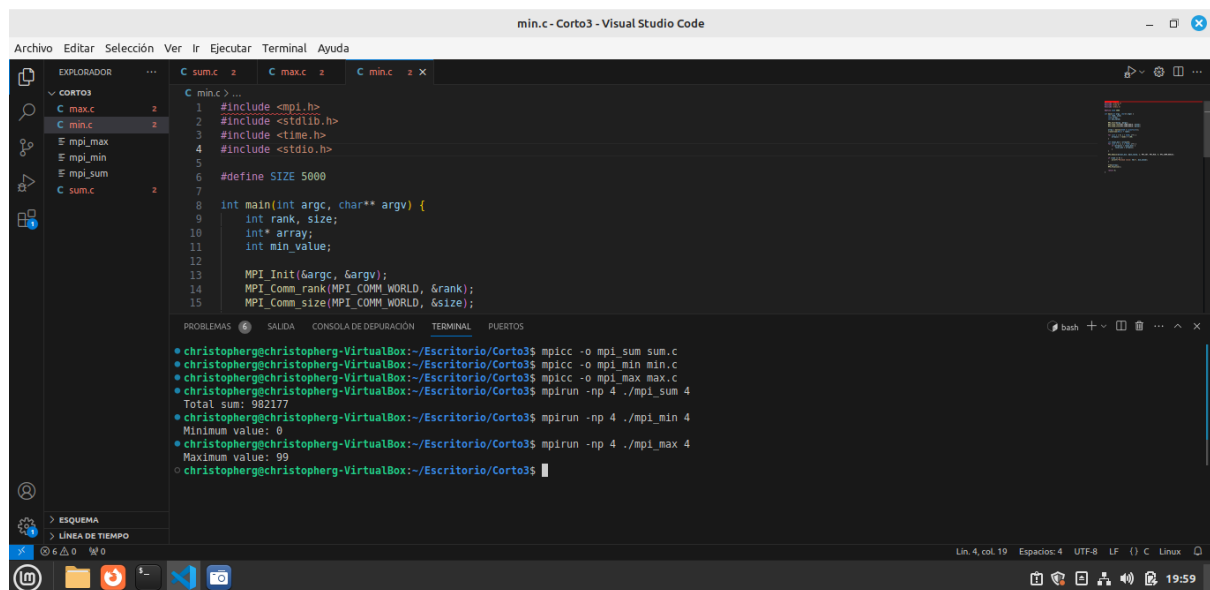
- Son instrucciones para llevar a cabo una tarea específica en las cuales se tiene acceso a lectura y escritura de información (más a nivel OS) sobre la red.

4. Liste y describa brevemente 4 subrutinas/directivas de Open MPI.

- MPI_COMM_WORLD() permite definir el espacio de red donde todo lo que se encuentre en su interior podrá recibir comunicación.
- MPI_Finalize(): Indica donde termina el entorno de comunicación
- MPI_Init(&argc, &argv): Inicializa todo el entorno donde se usará MPI
- MPI_Send(): Enviar mensaje a otro proceso
- MPI_Recv(): Recibir mensaje de otro proceso

Segunda Parte (30 puntos)

Cree un programa que, mediante el uso de Open MPI ejemplifique el uso de 3 operaciones de reducción sobre un array/vector de 5,000 elementos con valores enteros generados de manera aleatoria.



```
min.c - Corto3 - Visual Studio Code

Archivo Editar Selección Ver Ir Ejecutar Terminal Ayuda

EXPLORADOR
CORTO3
  C max.c
  C min.c
  E mpi_max
  E mpi_min
  E mpi_sum
  C sum.c

C min.c
1 #include <mpi.h>
2 #include <stdlib.h>
3 #include <time.h>
4 #include <stdio.h>
5
6 #define SIZE 5000
7
8 int main(int argc, char** argv) {
9     int rank, size;
10    int* array;
11    int min_value;
12
13    MPI_Init(&argc, &argv);
14    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
15    MPI_Comm_size(MPI_COMM_WORLD, &size);

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
christopher@christopher-VirtualBox:~/Escritorio/Corto3$ mpicc -o mpi_sum sum.c
christopher@christopher-VirtualBox:~/Escritorio/Corto3$ mpicc -o mpi_min min.c
christopher@christopher-VirtualBox:~/Escritorio/Corto3$ mpicc -o mpi_max max.c
christopher@christopher-VirtualBox:~/Escritorio/Corto3$ mpirun -np 4 ./mpi_sum 4
Total sum: 982177
christopher@christopher-VirtualBox:~/Escritorio/Corto3$ mpirun -np 4 ./mpi_min 4
Minimum value: 0
christopher@christopher-VirtualBox:~/Escritorio/Corto3$ mpirun -np 4 ./mpi_max 4
Maximum value: 99
christopher@christopher-VirtualBox:~/Escritorio/Corto3$
```

- mpicc -o mpi_sum sum.c
 - mpirun -np 4 ./mpi_sum 4
- mpicc -o mpi_min min.c
 - mpirun -np 4 ./mpi_min 4
- mpicc -o mpi_max max.c
 - mpirun -np 4 ./mpi_max 4

Tercera Parte (40 puntos)

Desarrolle en un análisis completo cuál sería su implementación de un programa en Open MPI que calcule la aproximación del resultado de una integral mediante las sumas de Riemann. Detalle cada una de las fases de su análisis. Coloque la descripción sobre los pasos a seguir y los eventos de comunicación presentes en el flujo de su implementación. Incluya los diagramas en la etapa/fase correspondiente. No es necesario que programe su implementación, aunque si lo ayuda a visualizar mejor el problema puede hacerlo.

Hint: Recuerde el ejercicio que hicimos en clase cuando platicamos sobre la comunicación punto a punto

Pseudocódigo en fases

- Fase 1: Inicializar valores (**r=0**)
 - El primer paso sería inicializar el entorno de Open MPI
 - El segundo paso sería definir en un método la función que se va a integrar recibiendo como parámetros a, b (los límites)
 - `funcionAIntegrar(a,b): {}`
- Fase 2: División de intervalos (**r=0**) (**Difusión**)
 - En esta fase habrá un pasó y será el cálculo de subintervalos
 - Cada proceso calcula su subintervalo local `[a_local, b_local]` en función de su rango y el número total de procesos.
- Fase 3: Cálculo de sumas parciales (**r=0-(n-1)**)
 - En esta fase cada proceso realiza la suma de Riemann en su subintervalo local y obtiene una suma parcial local.
- Fase 4: Reducción de las sumas parciales (**r=0**)
 - En esta fase, se realizará una reducción de todas las sumas parciales locales para obtener la suma total de Riemann. Se utiliza una operación de reducción de MPI (i.e, `MPI_Reduce`) para sumar todas las sumas parciales locales y obtener la suma total.
- Fase 5: Cálculo del resultado final (**r=0**) (**Recolección**)
 - Por último, en esta fase, el proceso 0 realiza el cálculo final de la integral sumando todas las sumas parciales reducidas y presentar un resultado final
- Fase 6: Finalización (**r=0**)
 - En esta fase, se liberan los recursos y se finaliza el entorno MPI.