



Multiprocesamiento en lenguaje C

*Introducción a MPI
Message Passing Interface*

Proyecto PAPIME PE104911

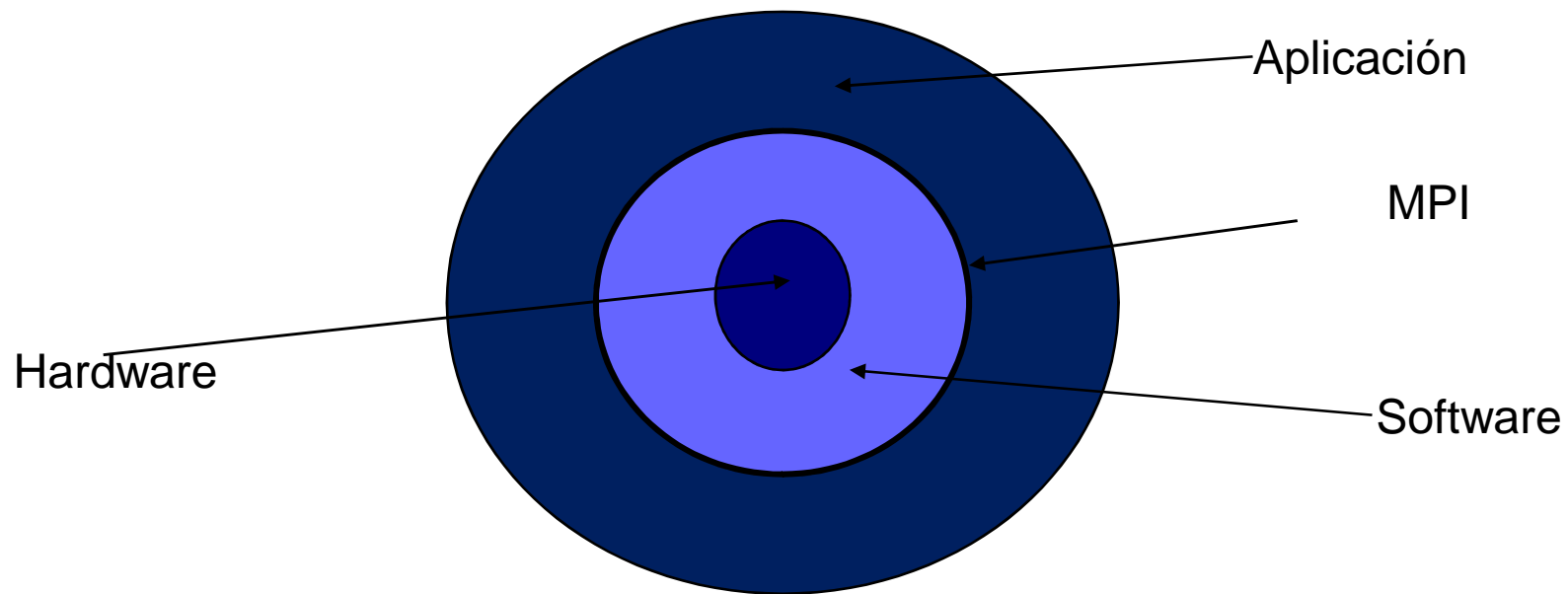
*Pertinencia de la enseñanza del cómputo
paralelo en el currículo de las ingenierías*



MPI

- MPI es un estándar de programación en paralelo mediante paso de mensajes
- Creado en 1993 como un estándar abierto por fabricantes y usuarios de sistemas paralelos.
- Biblioteca que incluye interfaces para FORTRAN, C , C++ y JAVA.
- Define **varias formas de comunicación**

Ubicación de MPI en el proceso de programación de aplicaciones paralelas





Modelo de programación

■ Hardware

- Memoria Distribuida
- Memoria Compartida (algunas implementaciones)
- Híbrido

■ Paralelismo explícito

■ MPI-1 Estático y MPI-2 Dinámico



MPI-1 VS MPI-2

- En el estandar MPI-1 se establece el numero de procesos a crear a momento de la ejecución
- En MPI-2
 - Se pueden crear procesos durante la ejecución
 - Se puede utilizar memoria de otro nodo. (operaciones de memoria remota)
 - Provee funciones dedicadas a E/S paralelas



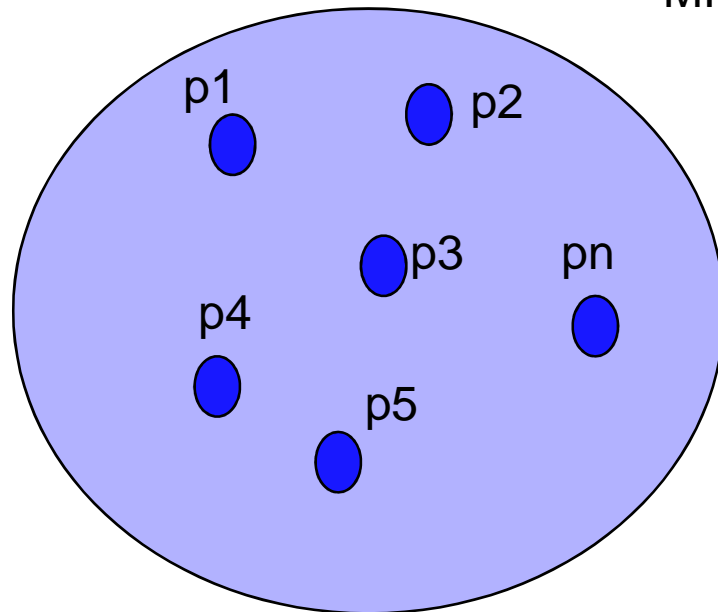
Funcionamiento

- La unidad básica en MPI son los **procesos**.
- Cada proceso se le asigna un **identificador** interno propio de MPI (**rank**).
- Tienen espacios de memoria independientes.
- Intercambio de información por **paso de mensajes**.

Comunicadores

- ¿Qué es un comunicador?

MPI_COMM_WORLD





Argumentos de las funciones en MPI

- **E-IN**: la función lo usa pero no lo actualiza.
- **S-OUT**: se puede actualizar este valor
- **E/S-INOUT**: se usa y lo actualiza

- Retorno de la función
 - MPI_SUCCESS



Funciones Básicas

- **MPI 1.2** tiene 129 funciones. Las funciones principales de MPI son:
 - MPI_Init
 - MPI_Finalize
 - MPI_Comm_size
 - MPI_Comm_rank
 - MPI_Send
 - MPI_Recv



C

```
int MPI_Init(int *argc, char *argv[]);
```

```
int MPI_Finalize(void);
```

```
int MPI_Comm_size(MPI_Comm comm, int *pcomm_size)
```

```
int MPI_Comm_rank(MPI_Comm comm, int *pcomm_rank)
```



Estructura de un programa en MPI

Incluir la biblioteca mpi .h

.

.

Inicializar ambiente MPI

.

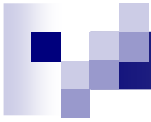
.

Cómputo y comunicaciones entre procesos

.

.

Terminar ambiente MPI



■ Ejemplo 1 primer programa



Comunicaciones

- Punto a punto
- Colectivas



Modelo de comunicación

- MPI define dos modelos de comunicación:
 - bloqueante (blocking)
 - no bloqueante (nonblocking).



Punto-a Punto

- básico(basic), **MPI_Send(...)**
- con buffer (buffered) **MPI_Bsend(...)**
- síncrono **MPI_Ssend(...)**
- listo (ready) **MPI_Rsend(...)**



Funciones Básicas de Comunicación

C

```
int MPI_Send(void *buf, int count, MPI Datatype dtype,  
             int dest,int tag, MPI Comm comm)
```

```
int MPI_Recv(void *buf, int count, MPI Datatype dtype,  
             int src, int tag, MPI Comm comm,  
             MPI Status *status)
```




Argumentos

buf: Dirección donde comienza el mensaje.

count: número de elementos del mensaje.

datatype: tipo del mensaje.

dest/source: posición relativa del proceso fuente/destino dentro del comunicador. **MPI_ANY_SOURCE:** permite recibir mensaje de cualquier fuente.

tag: etiqueta del mensaje.

MPI_ANY_TAG: cualquier etiqueta.

comm: comunicador del proceso.

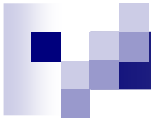
Para el estado

MPI_Status estado:

Proyecto PAPIME PE104911
Multiprocesamiento en lenguaje C
Elba Karen Sáenz García

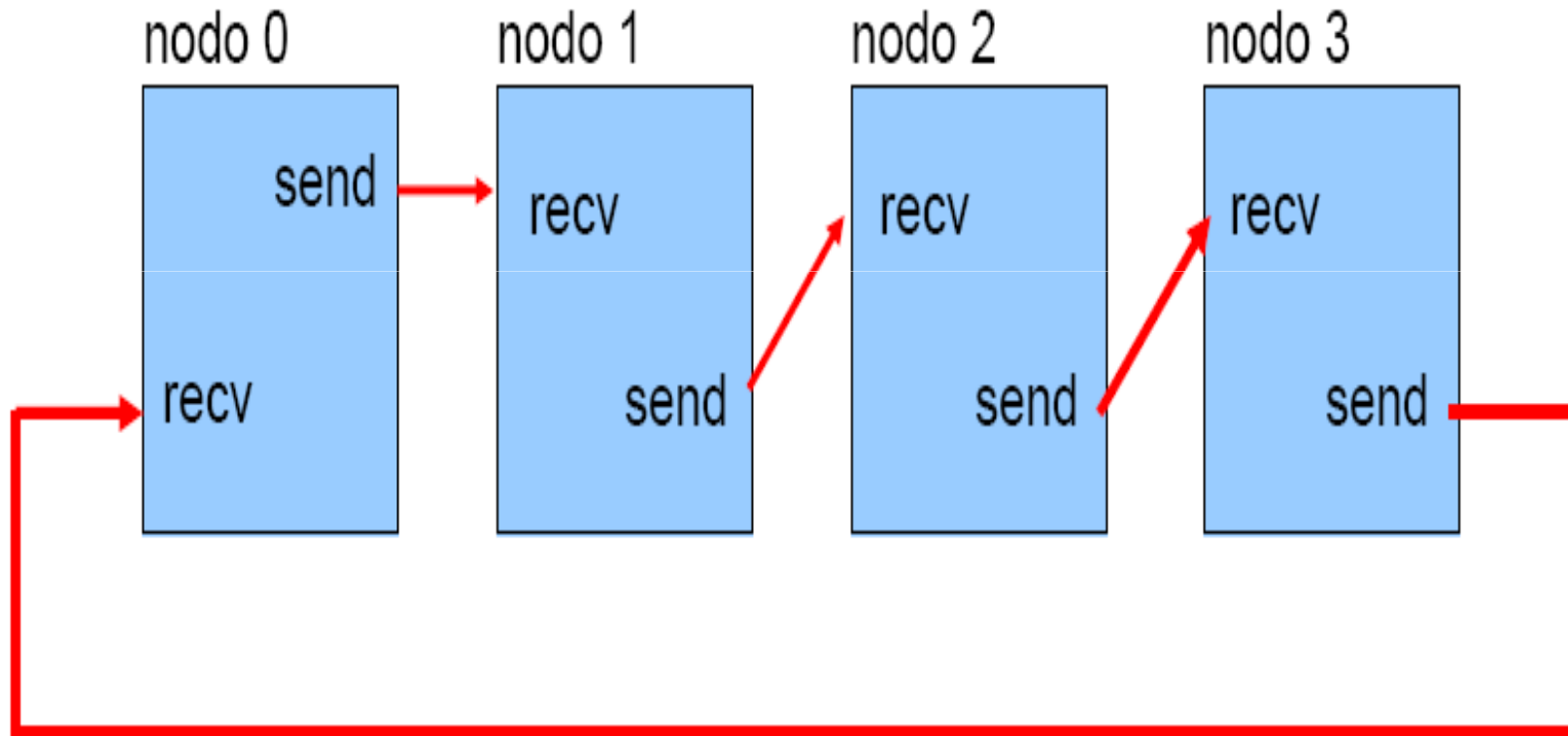
Tabla 1. Tipos de datos MPI

Tipos MPI	Tipos C equivalentes
MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_BYTE	Sin equivalente



- Ejemplo 2 de comunicación punto a punto
- Proceso 1 envíe un dato real al proceso 0, y el proceso 0 imprima el valor

Ejemplo 3





Deadlocks

- Un deadlock ocurre cuando un proceso queda esperando un mensaje que nunca recibirá.




Ejercicio 1

- Realizar un programa donde el proceso 0 asigne valores reales a un arreglo bidimensional de 10 renglones y 10 columnas y reparta la información entre otros 5 procesos diferentes, cada proceso imprimirá la parte que le enviaron.



Ejercicio 2

- Modificar el programa anterior para que los 5 procesos diferentes sumen una unidad a cada elemento de su sub-arreglo y regresen los datos al proceso 0 para actualizar los datos del arreglo.



```
rc=MPI_Init(&argc, &argv);
if(rc != MPI_SUCCESS)
{
printf("El programa termino\n");
MPI_Abort(MPI_COMM_WORLD,rc);
}
```




Comunicaciones Colectivas

- Comunicaciones realizadas entre un grupo de procesos los cuales están especificados en un comunicador
- Todos los procesos en el comunicador deben llamar la operación colectiva

Algunas funciones para comunicaciones colectivas

Process 0	Process 1'	Process 2	Process 3	Function Used	Process 0	Process 1'	Process 2	Process 3
a	b	c	d	<u>MPI_Gather</u>		a,b,c,d		
a	b	c	d	<u>MPI_Allgather</u>	a,b,c,d	a,b,c,d	a,b,c,d	a,b,c,d
	a,b,c,d			<u>MPI_Scatter</u>	a	b	c	d
a,b,c,d	e,f,g,h	ij,k,l	m,n,o,p	<u>MPI_Alltoall</u>	a,e, i,m	b,f, j,n	c,g, k,o	d,h, l,p
	b			<u>MPI_Bcast</u>	b	b	b	b
Send Buffer	Send Buffer	Send Buffer	Send Buffer		Receive Buffer	Receive Buffer	Receive Buffer	Receive Buffer



MPI_Bcast

- Envía datos de un proceso a todos los demás.

```
int MPI_Bcast ( void *buffer, int count, MPI_Datatype datatype, int root,  
MPI_Comm comm )
```

Buffer (Input/Output): Dirección de los datos

Count: Número de elementos en buffer

datatype: Tipo de datos

Root: Rank del proceso que contiene los datos que serán replicados

Comm: Comunicador



MPI_Gather

```
int MPI_Gather( void* sendbuf, int sendcount,  
               MPI_Datatype sendtype, void* recvbuf,  
               int recvcount, MPI_Datatype recvtype,  
               int root, MPI_Comm comm);
```



MPI_Scatter

- realiza la operación simétrica a MPI_Gather() .

```
int MPI_Scatter(void* sendbuf, int sendcount, MPI_Datatype sendtype,  
void* recvbuf, int recvcount, MPI_Datatype recvtype, int root,  
MPI_Comm comm);
```



Argumentos

Funciones de tranferencia de datos colectivas

ES	inbuf	dirección del <i>buffer</i> de entrada
E	incnt	número de elementos a enviar a cada uno
E	intype	tipo de dato de los elementos del <i>buffer</i> de entrada
S	outbuf	dirección del <i>buffer</i> de salida
E	outcnt	número de elementos a recibir de cada uno
E	outtype	tipo de dato de los elementos del <i>buffer</i> de salida
E	root	rango del proceso <i>root</i>
E	comm	comunicador



Operación de Reducción

■ MPI_Reduce

```
int MPI_Reduce(void *sendbuf,  
              void *recvbuf,  
              int count,  
              MPI_Datatype datatype,  
              MPI_Op op,  
              int root,  
              MPI_Comm comm)
```



MPI_Reduce

Tipos de Operaciones Para MPI_Reduce

Nombre C	Nombre C++	Operación
MPI_MAX	MPI::MAX	Busca el máximo
MPI_MIN	MPI::MIN	Busca el mínimo
MPI_SUM	MPI::SUM	Suma los valores
MPI_PROD	MPI::PROD	Multiplica los valores
MPI_LAND	MPI::LAND	And lógico
MPI_BAND	MPI::BAND	And por bits
MPI_LOR	MPI::LOR	Or lógico
MPI_BOR	MPI::BOR	Or por bits
MPI_LXOR	MPI::LXOR	Xor lógico
MPI_BXOR	MPI::BXOR	Xor por bits
MPI_MAXLOC	MPI::MAXLOC	Busca el máximo y su ubicación
MPI_MINLOC	MPI::MINLOC	Busca el mínimo y su ubicación



Ejemplo

- Realizar un programa que permita la creación de n procesos y cada uno de ellos aporte como dato su $id + 1$, y el proceso raíz sume esos valores parciales e imprime el resultado de la suma..




Pack y unpack

- Existen otras dos funciones en MPI que permiten empaquetar y desempaquetar datos
- `MPI_Pack`: Empaqueta datos en un buffer continuo en memoria para ser enviados como un mensaje único.
- `MPI_Unpack`: Desempaqueta los datos recibidos en un mensaje.



```
int MPI_Pack (void *inbuf, int incount,  
MPI_Datatype, datatype, void *outbuf, int  
outcount, int *position, MPI_Comm comm )
```

- Los tres primeros argumentos son información de los datos a empaquetar (dirección, número de elementos y tipo de datos).
- Los dos siguientes argumentos especifican la zona de memoria en que se van a empaquetar los datos (dirección y tamaño).



```
int MPI_Unpack (void *inbuf, int insize, int
    *position, void *outbuf, int outcount,
    MPI_Datatype datatype, MPI_Comm comm).
```

- Los dos primeros argumentos especifican los datos a desempaquetar (dirección y longitud).
- Los tres siguientes argumentos especifican los datos a desempaquetar (dirección, número de elementos y tipo de datos).



Ejemplo

El proceso con id 0 va a empaquetar un dato entero, uno real y un carácter y enviar la información empaquetada al proceso 1 , quien la desempaquetará y mostrará los datos que recibió.



Ejemplo de aplicación

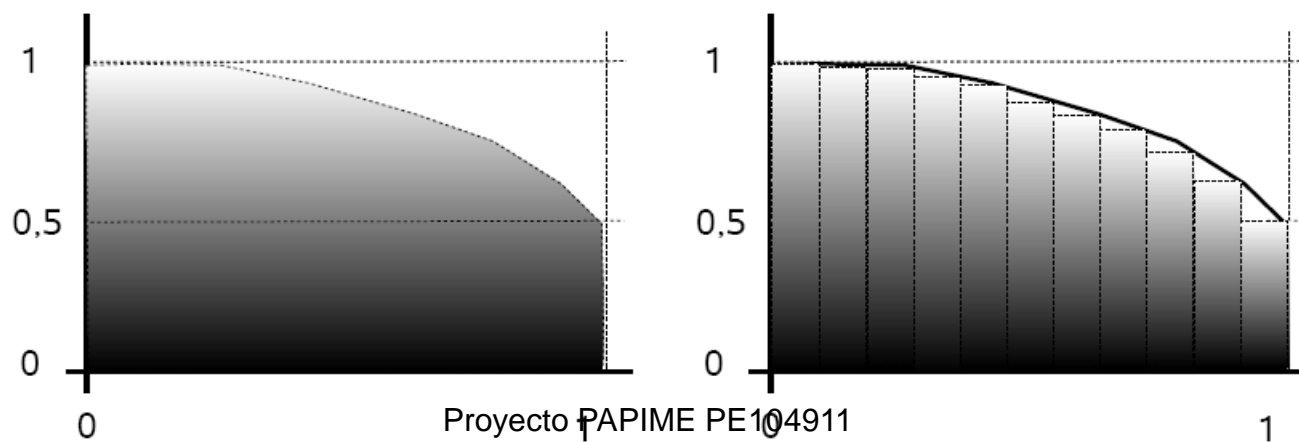
Proyecto PAPIME PE104911

Pertinencia de la enseñanza del cómputo paralelo en el currículo de las ingenierías

Cálculo del número Pi

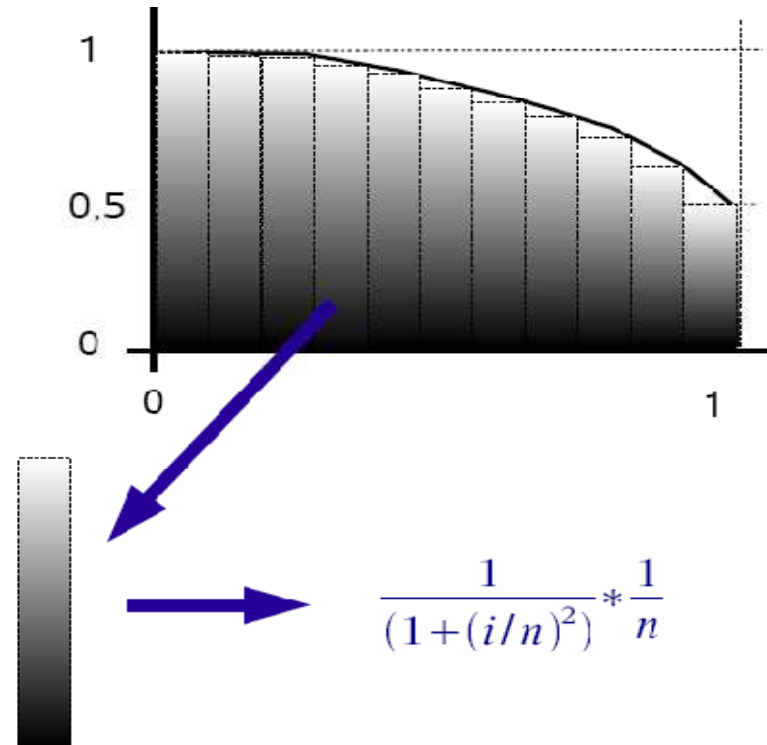
- Calcular el valor del número pi utilizando la aproximación a la integral que se muestra en la siguiente ecuación.

$$\int_{x=0}^1 \frac{1}{1+x^2} dx = \frac{\pi}{4} \approx \sum_{i=0}^n \frac{1}{(1+(i/n)^2)} * \frac{1}{n}$$



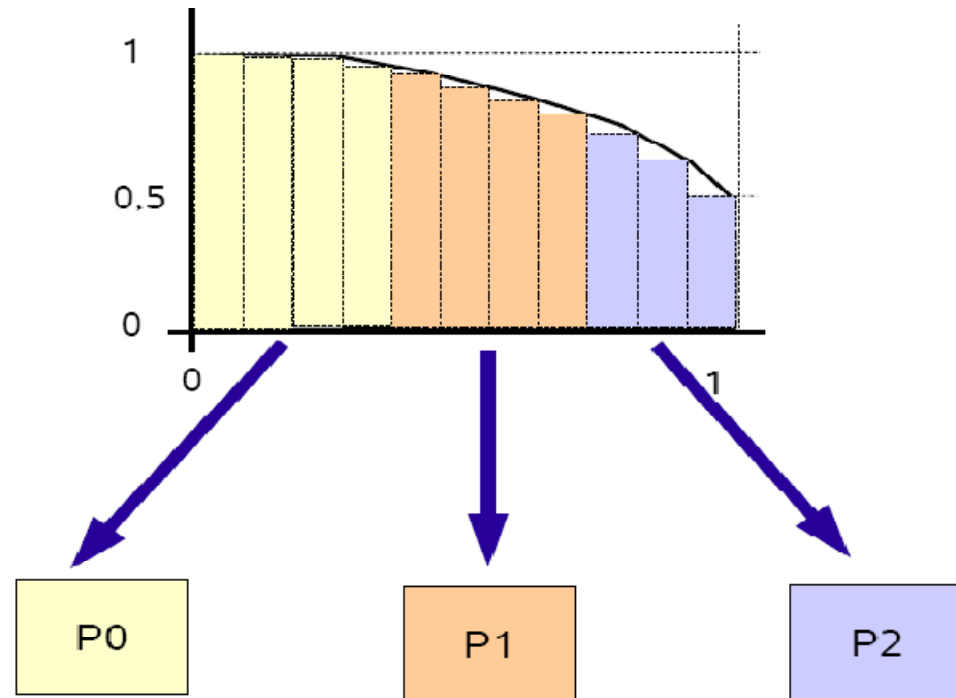
Descomponer el problema

- Dividir en varios rectángulos
- Cálculo del área de un rectángulo
- Suma de todas las áreas
- Mismo cálculo, sobre diferentes rectángulos



Asignando un número de tareas a procesos

- Proceso divide
- Proceso reparte
- Proceso calculan
- Proceso colecta resultados





Realización código paralelo

■ Partiendo del código secuencial

```
void main(int argc, char *argv[])
    int i, n = 1000;
    double h, pi, x;

    h = 1.0 / (double) n;
    pi = 0.0;
    for (i=0; i<n; i++){
        x = (double) i / (double) n;
        pi += (1.0 / (1.0 + x*x)) * h;
    }
    printf("Pi es aproximadamente %22.16e\n", 4.0*pi);
}
```



Actividad

- **Realizar el código paralelo.**
- **Además explicar el funcionamiento de la función MPI_Reduce en el código.**
- **Subir su código fuente.**



Grupos y Comunicadores en MPI

Proyecto PAPIME PE104911

*Pertinencia de la enseñanza del cómputo
paralelo en el currículo de las ingenierías*



Grupos y Comunicadores

- MPI puede definir comunicadores con un número menor de procesos.
- El programador puede crear un grupo(de procesos) y asociarlo a un comunicador
- El nuevo comunicador puede ser usado en comunicaciones punto a punto y colectivas.
- Los grupos y comunicadores son objetos que el programador utiliza mediante el llamado manejador(handle)



Grupos y Comunicadores

- El manejador(handle) para el comunicador que incluye a todos los procesos es `MPI_COMM_WORLD`
- Grupo: Es una colección ordenada de procesos a los que se les asocia un id entre 0 y $n-1$, n es el numero de procesos en ese grupo.



Consideraciones y Restricciones

Grupos/Comunicadores(son dinámicos).
Pueden ser creados y destruidos durante la ejecución.

Un proceso puede pertenecer o uno o más Grupos/Comunicadores.

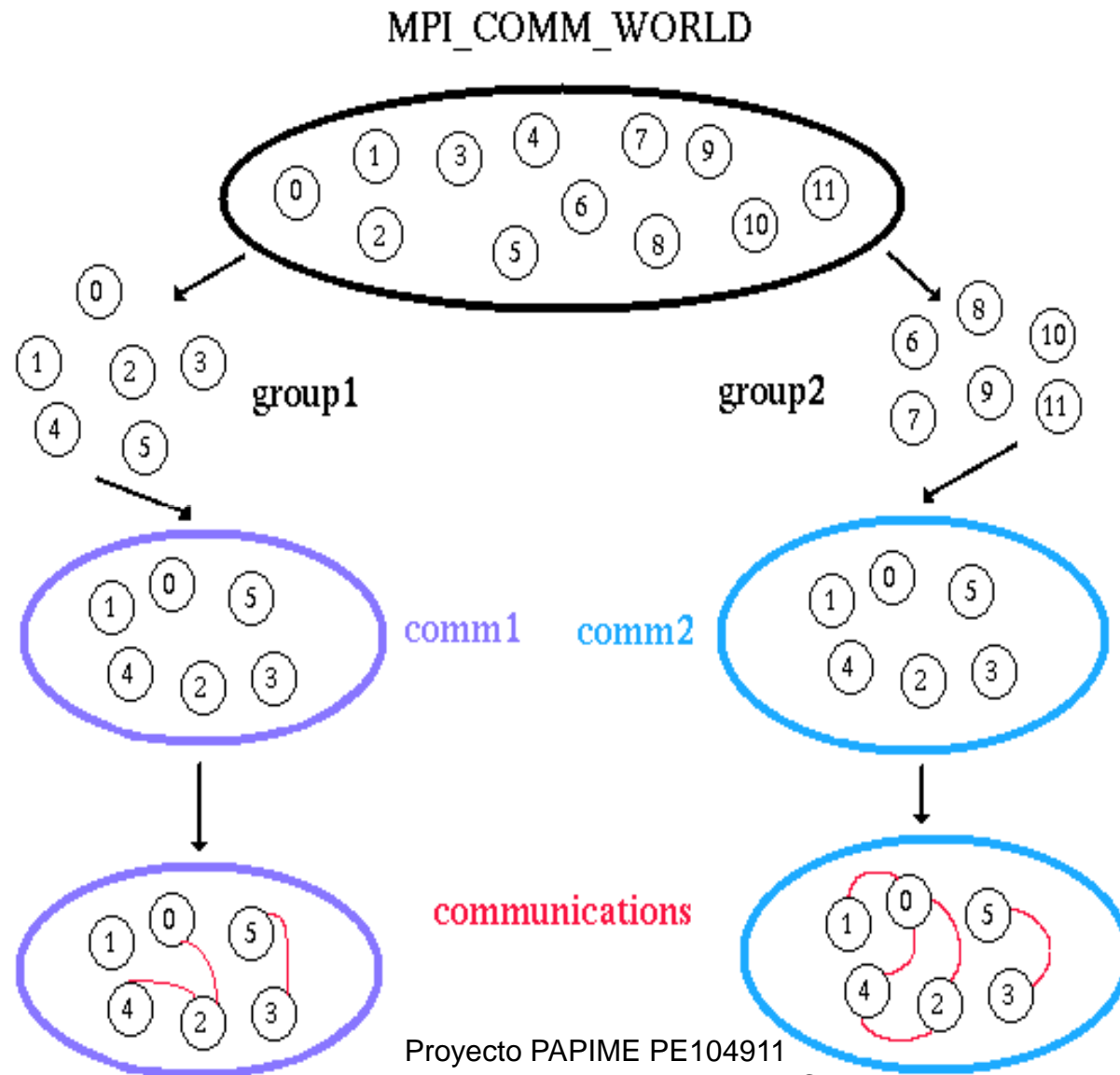
Se tienen 40 funciones relacionadas a Grupos, Comunicadores y Topologías Virtuales



Uso Común

1. Obtener el manejador de grupo global de MPI_COMM_WORLD, utilizando **MPI_Comm_group**
2. Formar un nuevo grupo con **MPI_Group_incl**
3. Crear nuevo comunicador con el nuevo grupo utilizando **MPI_Comm_create**
4. Determinar el rank en el nuevo comunicador, con **MPI_Comm_rank**
5. Realizar Comunicación utilizando las funciones de MPI
6. Destruir Grupos/Comunicadores (Opcional)

Ejemplo





Ejercicio

- Realizar un programa que cree 2 comunicadores.
- Donde:
 - En el primer comunicador se calcule el numero pi.
 - En el segundo, el nuevo proceso cero envíe un arreglo bidimensional de 5x3 a todos los demás.