Programación paralela y concurrente (TC2025)

Tema 6. Programación multinúcleo

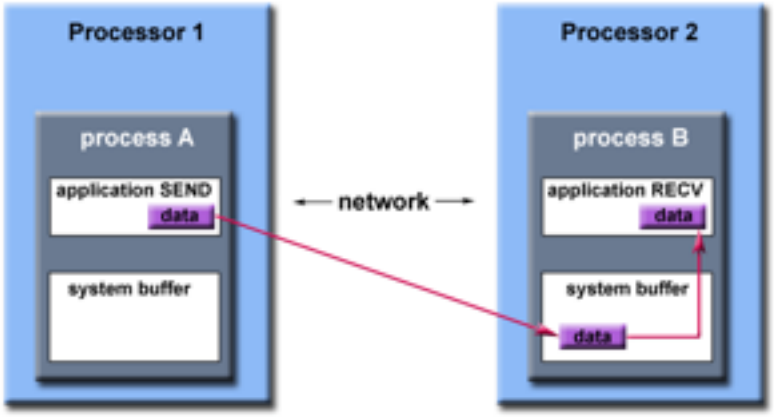
Instituto Tecnológico y de Estudios Superiores de Monterrey. Campus Cuernavaca
Departamento de Tecnologías de Información y Mecatrónica
Dr. Vicente Cubells (vcubells@itesm.mx)

Temario

- Conceptos generales de paso de mensajes
- Introducción a MPI
- Modelo de programación MPI
- Comunicadores
- Algunas funciones
- Ejemplos

Conceptos generales…

- Programación SPMD y MPMD
- Send() y Receive()
- Operaciones síncronas/asíncronas
  - Bloqueos en operaciones síncronas
- Operaciones con buffer/sin buffer

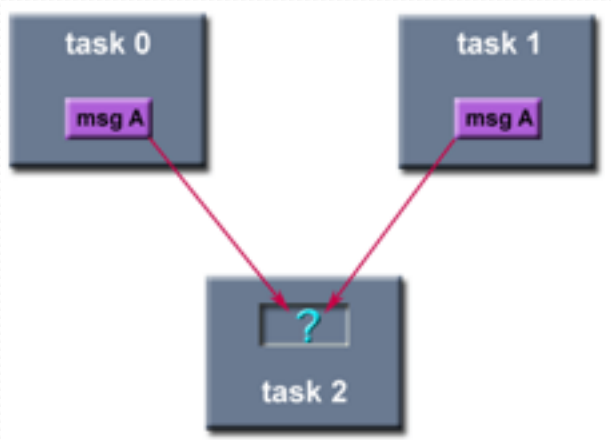


Path of a message buffered at the receiving process

Operaciones con buffer susceptibles a bloqueos

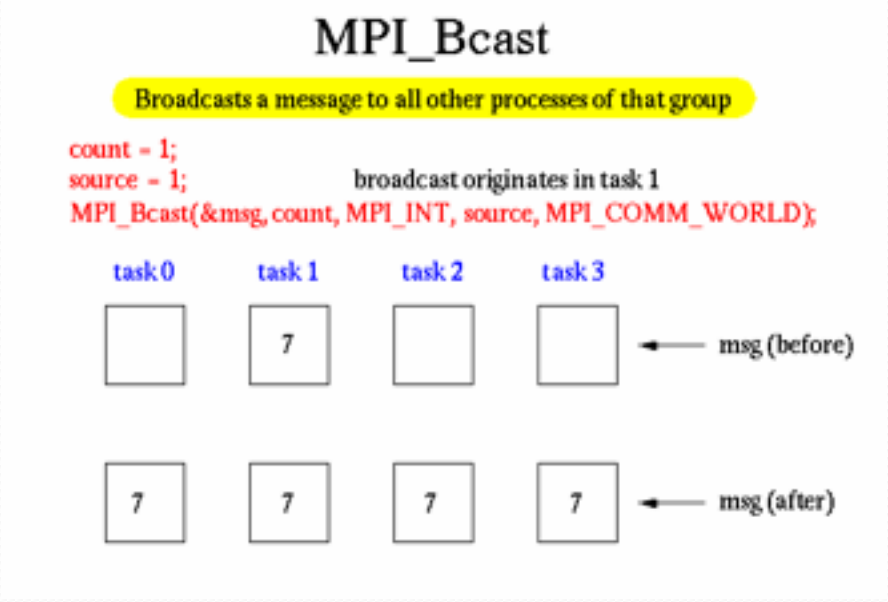
Conceptos generales…

- Operaciones bloqueantes / no bloqueantes
- Recepción ordenada / desordenada



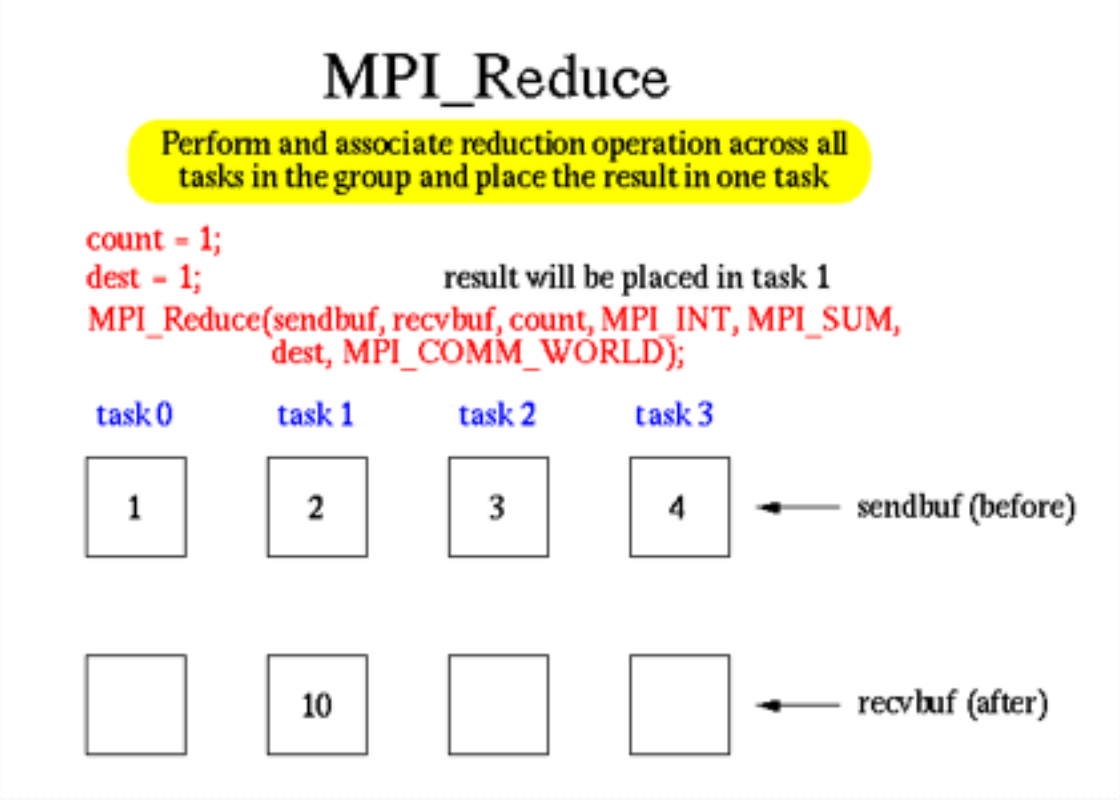
Conceptos generales…

- Sincronizaciones
  - Barreras
- Broadcast 1..N (Mismo mensaje)
  - Se puden utilizar jerarquías, mallas, hipercubos, …



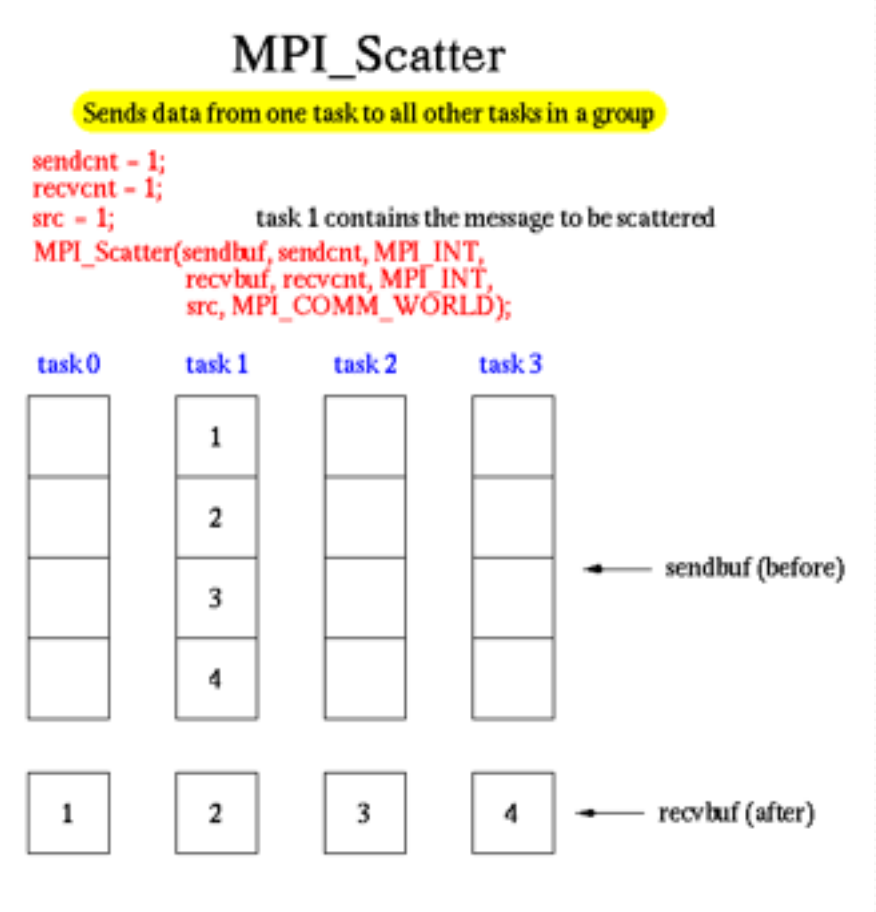
Conceptos generales…

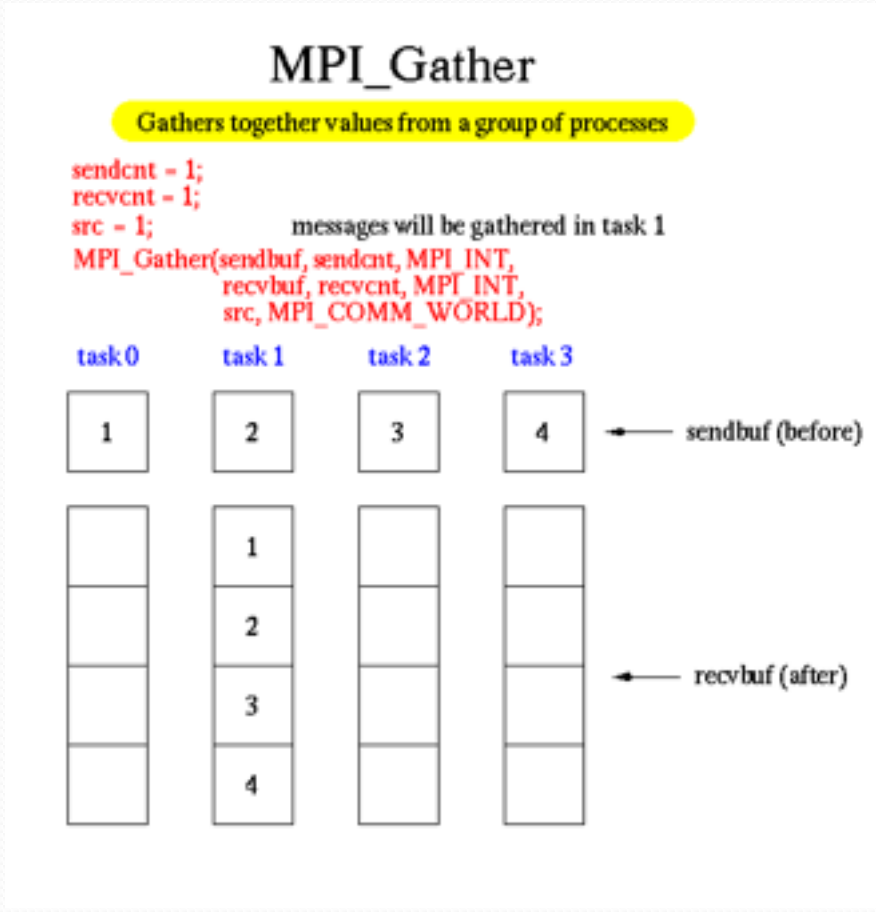
- Reducciones N..1



Conceptos generales…

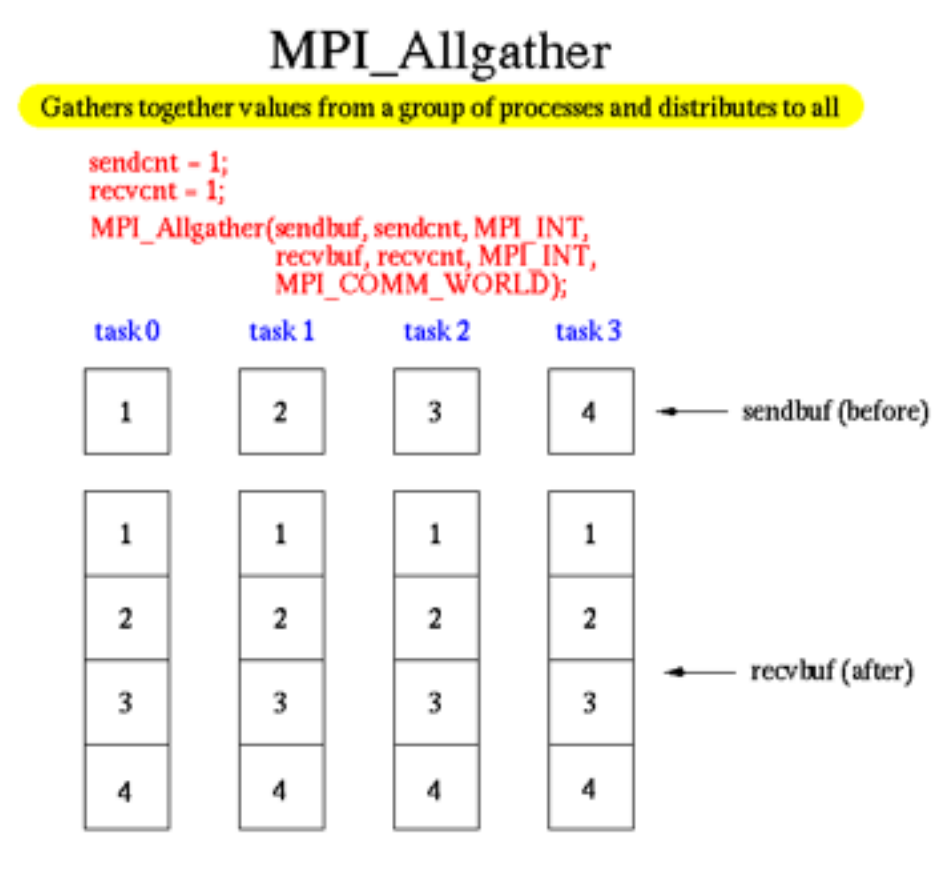
- Scatter/Gatter (Mensajes diferentes)





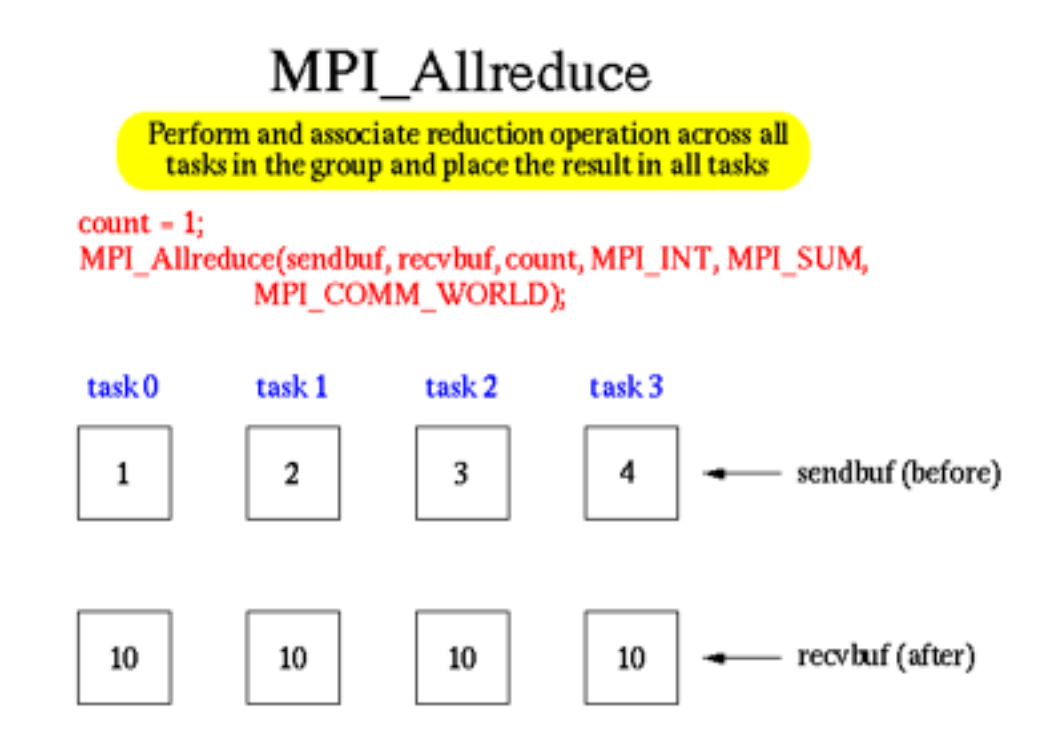
Conceptos generales…

- Broadcast N..N (Mismo mensaje a todos)



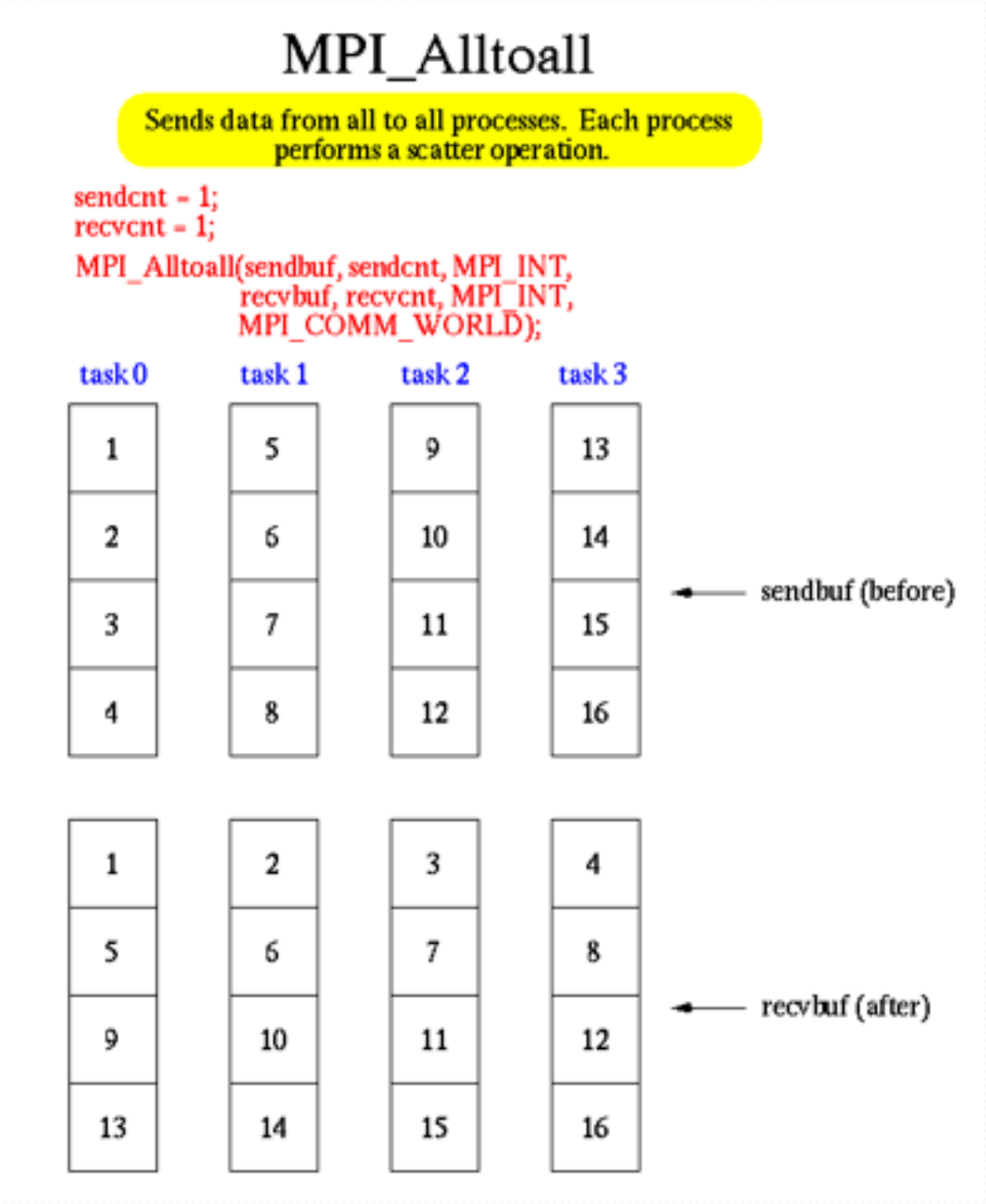
Conceptos generales…

- Reducciones N..N



Conceptos generales

- Intercambio total N..N (Mensajes diferentes)



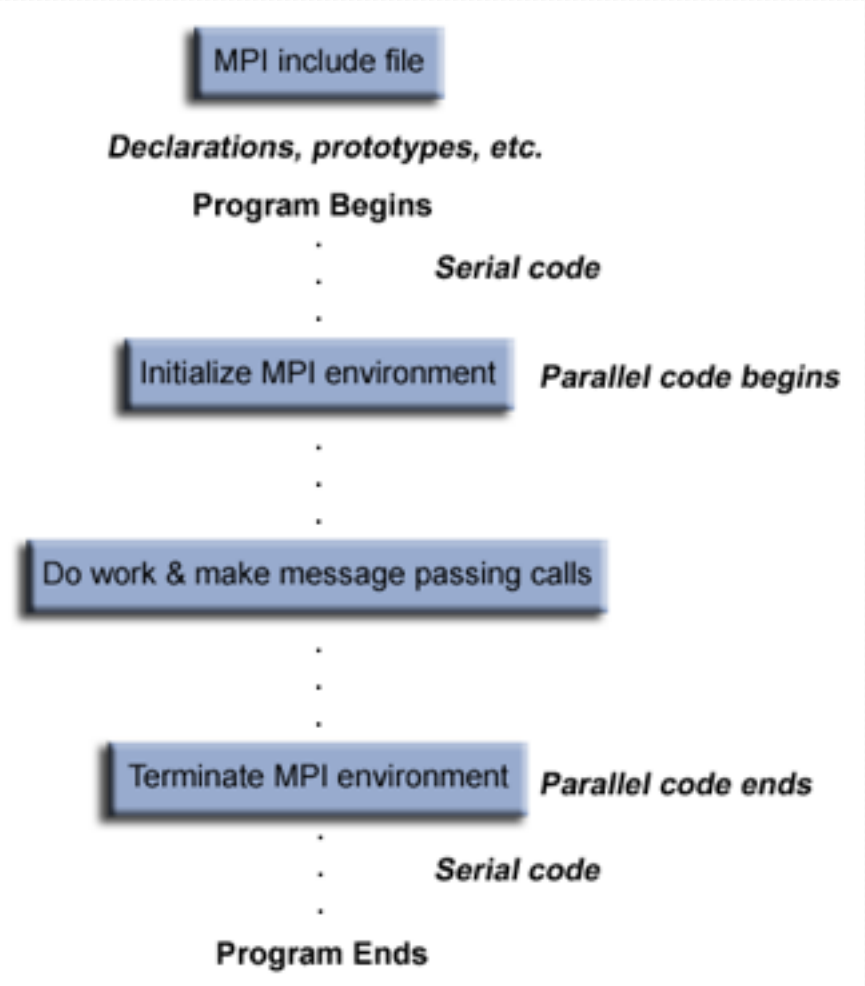
¿Qué es MPI?

- Es una especificación no una implementación particular
- No es una manera revolucionaria de escribir programas paralelos, sino un intento por agrupar las mejores características de las librerías de paso de mensajes
- Ofrece:
  - Estandarización: No ANSI, si por consenso
  - Portabilidad: No debe haber necesidad de cambiar el código
  - Rendimiento: Depende de las implementaciones y el hardware
  - Funcionalidad: mas de 120 funciones
  - Disponibilidad: varias plataformas

Modelo de programación

- Plataformas de hardware que soporta:
  - Para arquitecturas de memoria distribuida
  - En arquitecturas de memoria compartida como SMP / NUMA
  - Híbridos: clusters, redes heterogénas, …
- Paralelismo explícito

Estructura de un programa MPI



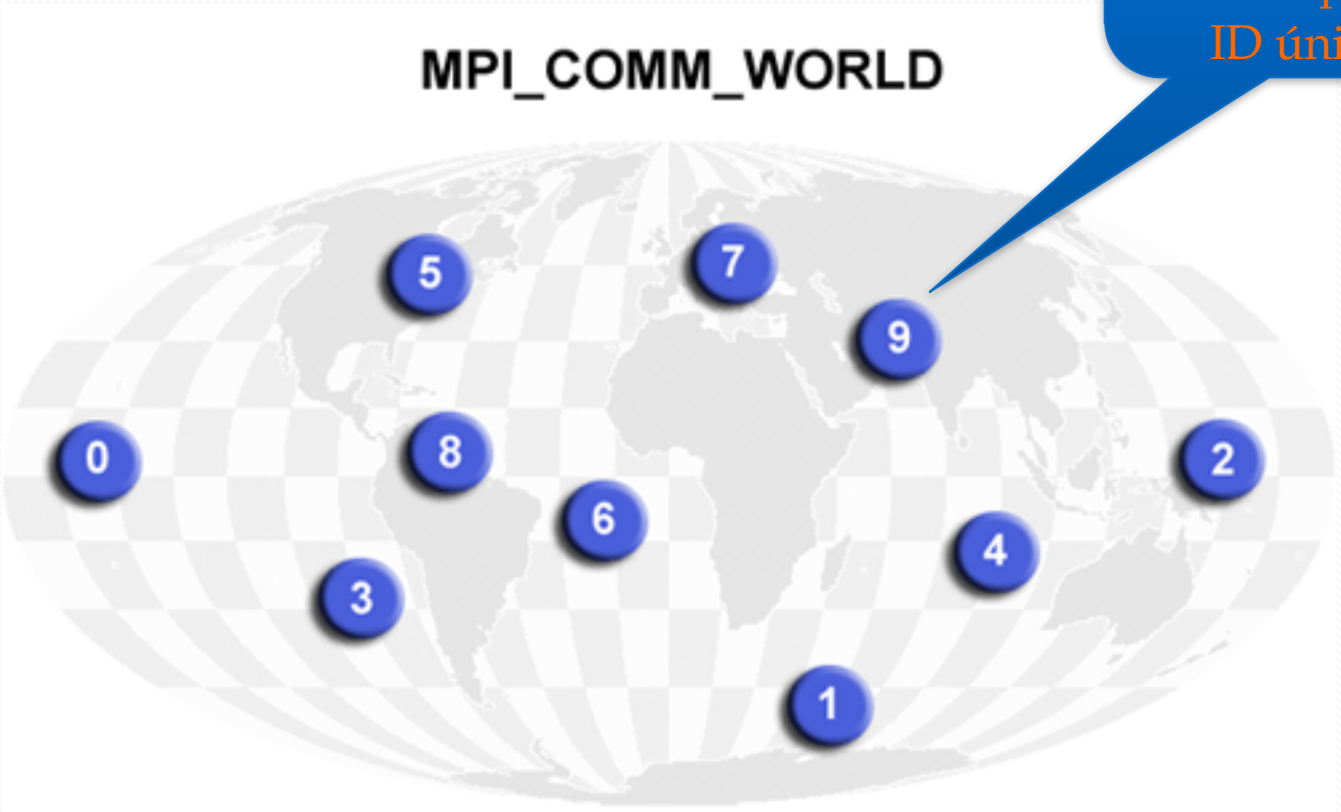
rc = MPI_Xxxx(parameter, ...)

Ejemplo:

MPI_Init (&argc,&argv)

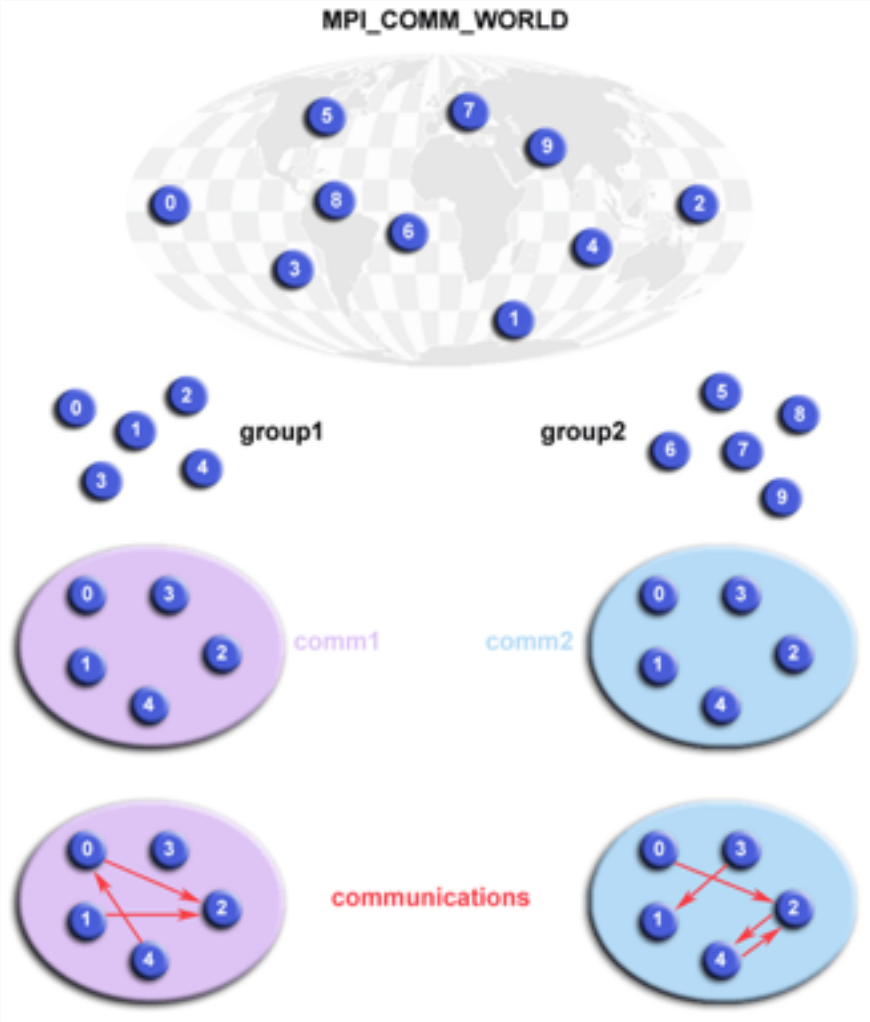
rc es el código de error
MPI_SUCCESS si no hubo error

Comunicadores…



Dentro de un comunicador cada proceso tiene su propio ID único

Comunicadores



Dentro de un comunicador cada proceso tiene su propio ID único

Funciones básicas

MPI_Init (&argc,&argv)

MPI_Initialized (&flag)

MPI_Comm_size (comm,&size)

MPI_Comm_rank (comm,&rank)

MPI_Finalize ()

MPI_Abort (comm,errorcode)

```
#include "mpi.h"
#include <stdio.h>

int main(argc,argv)
int argc;
char *argv[]; {
    int  numtasks, rank, rc;

    rc = MPI_Init(&argc,&argv);
    if (rc != MPI_SUCCESS) {
        printf ("Error al iniciar MPI \n");
        MPI_Abort(MPI_COMM_WORLD, rc);
    }

    MPI_Comm_size(MPI_COMM_WORLD,&numtasks);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    printf ("Soy la tarea %d de %d\n", rank,
           numtasks);

    /*******  hacer algo *******/

    MPI_Finalize();
}
```

Argumentos de las funciones

MPI_Send(buffer,count,type,dest,tag,comm)

MPI_Recv(buffer,count,type,source,tag,comm,status)

Bloqueado

MPI_Isend(buffer,count,type,dest,tag,comm,request)

MPI_Irecv(buffer,count,type,source,tag,comm,request)

No Bloqueado

MPI_CHAR, MPI_INT, MPI_LONG, MPI_FLOAT, MPI_DOUBLE

Funciones bloqueantes

MPI_Send (&buf,count,datatype,dest,tag,comm)

MPI_Recv (&buf,count,datatype,source,tag,comm,&status)

MPI_Ssend (&buf,count,datatype,dest,tag,comm)

MPI_Bsend (&buf,count,datatype,dest,tag,comm)

MPI_Buffer_attach (&buffer,size)
MPI_Buffer_detach (&buffer,size)

MPI_Wait (&request,&status)

MPI_Probe (source,tag,comm,&status)

Ejemplo Ping-Pong

```
#include "mpi.h"
#include <stdio.h>

int main(argc,argv)
int argc;
char *argv[];  {
    int numtasks, rank, dest, source, rc, count, tag=1;
    char inmsg, outmsg='x';
    MPI_Status Stat;

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    if (rank == 0) {
        dest = 1;
        source = 1;
        rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
        rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
    }

    else if (rank == 1) {
        dest = 0;
        source = 0;
        rc = MPI_Recv(&inmsg, 0, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
        rc = MPI_Send(&outmsg, 0, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
    }

    rc = MPI_Get_count(&Stat, MPI_CHAR, &count);
    printf("Task %d: Received %d char(s) from task %d with tag %d \n",
           rank, count, Stat.MPI_SOURCE, Stat.MPI_TAG);


    MPI_Finalize();
}
```

Funciones no bloqueantes

MPI_Isend (&buf,count,datatype,dest,tag,comm,&request)

MPI_Irecv (&buf,count,datatype,source,tag,comm,&request)

MPI_Issend (&buf,count,datatype,dest,tag,comm,&request)

MPI_Ibsend (&buf,count,datatype,dest,tag,comm,&request)

MPI_Test (&request,&flag,&status)

MPI_Iprobe (source,tag,comm,&flag,&status)

Ejemplo paso de un token

```
#include "mpi.h"
#include <stdio.h>

int main(argc,argv)
int argc;
char *argv[];  {
    int numtasks, rank, next, prev, buf[2], tag1=1, tag2=2;
    MPI_Request reqs[4];
    MPI_Status stats[2];

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    prev = rank-1;
    next = rank+1;
    if (rank == 0)  prev = numtasks - 1;
    if (rank == (numtasks - 1))  next = 0;

    MPI_Irecv(&buf[0], 1, MPI_INT, prev, tag1, MPI_COMM_WORLD, &reqs[0]);
    MPI_Irecv(&buf[1], 1, MPI_INT, next, tag2, MPI_COMM_WORLD, &reqs[1]);

    MPI_Isend(&rank, 1, MPI_INT, prev, tag2, MPI_COMM_WORLD, &reqs[2]);
    MPI_Isend(&rank, 1, MPI_INT, next, tag1, MPI_COMM_WORLD, &reqs[3]);


    MPI_Waitall(4, reqs, stats);

    MPI_Finalize();
}
```

Funciones de comunicación colectiva

MPI_Barrier (comm)

MPI_Bcast (&buffer,count,datatype,root,comm)

MPI_Reduce (&sendbuf,&recvbuf,count,datatype,op,root,comm)

MPI_Scatter (&sendbuf,sendcnt,sendtype,&recvbuf,
recvcnt,recvtype,root,comm)

MPI_Gather (&sendbuf,sendcnt,sendtype,&recvbuf,
recvcount,recvtype,root,comm)

MPI_Allgather (&sendbuf,sendcount,sendtype,&recvbuf,
recvcount,recvtype,comm)

MPI_Allreduce (&sendbuf,&recvbuf,count,datatype,op,comm)

MPI_Alltoall (&sendbuf,sendcount,sendtype,&recvbuf,
recvcnt,recvtype,comm)

Ejemplo de scatter

```
#include "mpi.h"
#include <stdio.h>
#define SIZE 4

int main(argc,argv)
int argc;
char *argv[];  {
    int numtasks, rank, sendcount, recvcount, source;
    float sendbuf[SIZE][SIZE] = {
        {1.0, 2.0, 3.0, 4.0},
        {5.0, 6.0, 7.0, 8.0},
        {9.0, 10.0, 11.0, 12.0},
        {13.0, 14.0, 15.0, 16.0}  };
    float recvbuf[SIZE];

    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);

    if (numtasks == SIZE) {
        source = 1;
        sendcount = SIZE;
        recvcount = SIZE;
        MPI_Scatter(sendbuf,sendcount,MPI_FLOAT,recvbuf,recvcount,
                    MPI_FLOAT,source,MPI_COMM_WORLD);

        printf("rank= %d  Results: %f %f %f %f\n",rank,recvbuf[0],
               recvbuf[1],recvbuf[2],recvbuf[3]);
    }
    else
        printf("Must specify %d processors. Terminating.\n",SIZE);

    MPI_Finalize();
}
```