



Programación Paralela - CC332

2021-I

José Fiestas

15/06/21

Universidad Nacional de Ingeniería
jose.fiestas@uni.edu.pe

Unidad 4: Comunicación y coordinación

Objetivos:

1. Pasos de Mensaje: MPI, Mensajes Punto a Punto, MPI, Comunicación Colectiva, Blocking vs non-blocking
2. Comunicacion Global, topologias
3. Memoria Compartida: OMP, Constructores y cláusulas, CUDA, optimizacion con GPUs
4. Programacion Hibrida

Message Passing Interface

- Facilita operaciones colectivas de comunicación
- No interfiere con comunicación punto a punto
- Puede o no sincronizar procesos
- El buffer se re-utiliza solo cuando el proceso termina
- Todos los procesos en un comunicador participan de comunicación colectiva

MPI_Barrier:

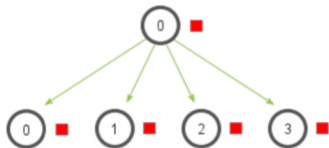
Operación de sincronización. No utiliza data. Funciona bloqueando el proceso de llamada hasta que todos los miembros del grupo han llamado a la operación

MPI_Barrier(comm)

MPI_Bcast:

Se envía mensaje del rank
'root' a todos los
procesos en el grupo

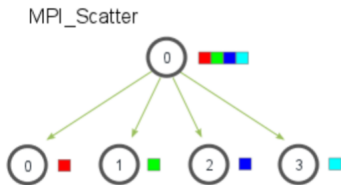
MPI_Bcast



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

MPI_Scatter:

Distribuye mensajes de una sola fuente a cada proceso en el grupo



MPI_Scatter(&buf,sendcnt,sendtype,&recvbuf,recvnt,recvtype,root,comm)

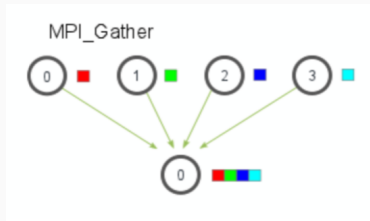
- **&buf**: dirección del buffer que reside en el nodo raíz
- **sendcnt**: número de elementos a ser enviados por proceso (elementos del array / número de nodos)
- **sendtype**: tipo de elementos enviados
- **&recvbuf**: buffer que recibe **recvnt** elementos del tipo **recvtype**
- **root**: nodo raíz
- **comm**: comunicador en el que residen los procesos

MPI_Gather:

Recopila información de cada proceso en un solo destino.

Reverso de MPI_Scatter.

Los elementos están ordenados de acuerdo al rango del proceso de donde son recibidos



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

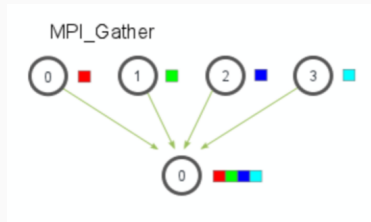
- **&sendbuf**: dirección del buffer que reside en el nodo raíz
- **sendcnt**: número de elementos a ser enviados por proceso
- **sendtype**: tipo de elementos enviados
- **&recvbuf**: buffer que recibe **recvnt** elementos del tipo **recvtype**
- **root**: nodo raíz
- **comm**: comunicador en el que residen los procesos

MPI_Allgather:

Recopila información de cada proceso en todos los demás procesos. Actúa como un MPI_Gather seguido de un MPI_Bcast

Los elementos son recopilados en el orden de rango de donde provienen

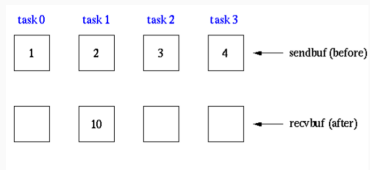
No contiene un proceso principal (root)



```
MPI_Allgather(&sendbuf,sendcnt,sendtype,&recvbuf,recvnt,  
recvtype,comm)
```

MPI_Reduce:

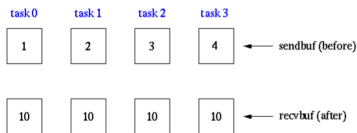
Recopila los datos del grupo y los pone en un task



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

MPI_Allreduce:

Recopila los datos del grupo y los pone en todos los task del grupo



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

Operadores pre-definidos






MPI Name	Function
MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_SUM	Sum
MPI_PROD	Product
MPI LAND	Logical AND
MPI_BAND	Bitwise AND
MPI_LOR	Logical OR
MPI_BOR	Bitwise OR
MPI_LXOR	Logical exclusive OR
MPI_BXOR	Bitwise exclusive OR
MPI_MAXLOC	Maximum & location
MPI_MINLOC	Minimum & location

Ejemplo 01: Bcast y reducción

```
1  int main(int argc, char*argv[])
2  {
3      MPI_Init(&argc,&argv);
4      MPI_Comm_rank(MPI_COMM_WORLD, &rank);
5      MPI_Comm_size(MPI_COMM_WORLD, &size);
6      ...
7      MPI_Bcast(&data,1,MPI_INT,0,MPI_COMM_WORLD);
8      ...
9      MPI_Reduce(&data,&res,DATA_SIZE,MPI_INT,MPI_SUM,0,MPI_COMM_WORLD);
10     ...
11     if(me==0){
12         // imprimir resultados en maestro
13     }
14     MPI_Finalize();
15 }
```

Ejemplo 02: Scatter/Gather

```
1  int main(int argc, char*argv[])
2  {
3      MPI_Init(&argc,&argv);
4      MPI_Comm_rank(MPI_COMM_WORLD, &rank);
5      MPI_Comm_size(MPI_COMM_WORLD, &size);
6      ...
7      if(rank==0){
8          //inicializar array globaldata[np] en proceso 0
9      }
10     MPI_Scatter(&globaldata,1,MPI_INT,&localdata,1,MPI_INT,0,MPI_COMM_WORLD);
11     ...
12     // multiplicar localdata por 2
13     MPI_gather(&localdata,1,MPI_INT,&globaldata,1,MPI_INT,0,MPI_COMM_WORLD);
14     ...
15     if(me==0){
16         // imprimir globaldata en maestro
17     }
18     ...
19     MPI_Finalize();
20 }
```

-  David B. Kirk and Wen-mei W. Hwu *Programming Massively Parallel Processors: A Hands-on Approach*. 2nd. Morgan Kaufmann, 2013. isbn: 978-0-12-415992-1.
-  Norm Matloff. *Programming on Parallel Machines*. University of California, Davis, 2014.
-  Peter S. Pacheco. *An Introduction to Parallel Programming*. 1st. Morgan Kaufmann, 2011. isbn: 978-0-12-374260- 5.
-  Michael J. Quinn. *Parallel Programming in C with MPI and OpenMP*. 1st. McGraw-Hill Education Group, 2003. isbn: 0071232656.
-  Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Program- ming*. 1st. Addison-Wesley Professional, 2010. isbn: 0131387685, 9780131387683.