

IMT2112 - Algoritmos Paralelos en Computación Científica

Comunicación con MPI

Elwin van 't Wout

1 de octubre de 2019



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

Facultad de Matemáticas • Escuela de Ingeniería

imc.uc.cl

Clase previa

- La programación en memoria distribuida
- La biblioteca MPI

Agenda

- ¿Como comunicar entre procesos con MPI?

La programación distribuida

Sección 2.6.3 del libro de Eijkhout

Parte 1 del libro “Parallel Programming in MPI and OpenMP” de Eijkhout

Comunicación proceso a proceso

Comunicación	Enviar x a p_1	Recibir x de p_0	Remark
Unilateral (<i>one-sided</i>)	Poner x en el espacio de datos de p_1	Extraer x del espacio de datos de p_0	Rápido y peligroso
Bilateral (<i>two-sided</i>)			Requiere dos instrucciones
a) sin bloqueo	Etiquetar x para enviar a p_1 y continuar	Continuar mientras esperar de recolectar hasta que p_0 haya etiquetado x para enviar	Se puede usar barreras para sincronizar
b) con bloqueo	Etiquetar x para enviar a p_1 y esperar recibir confirmación	Esperar hasta que p_0 haya etiquetado x para ser enviado, recójalo y envíe la confirmación	Lento y confiable; posibilidad de un <i>deadlock</i>

Comunicación en MPI

- Enviar con bloqueo (*blocking send*)
 - header

```
int MPI::Send(const void* buf, int count, MPI_Datatype  
datatype, int dest, int tag, MPI_Comm comm)
```
 - ejemplo

```
MPI::Send(&sendData, 1, MPI_DOUBLE, 1, 0,  
MPI_COMM_WORLD)
```

Comunicación en MPI

- Recibir con bloqueo (*blocking receive*)
 - header

```
int MPI::Recv(void* buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)
```
 - ejemplo

```
MPI::Recv(&receiveBuffer, 2, MPI_FLOAT, 0, 10,  
MPI_COMM_WORLD, &status)
```
 - Se puede inicializar un objeto `MPI_Status` como:
`MPI_Status status;`

Comunicación en MPI

- Operaciones colectivas
 - comunicación con bloque en todos los procesos
- *Broadcast*
 - header

```
int MPI::Bcast(void* buffer, int count, MPI_Datatype datatype,
               int root, MPI_Comm comm)
```
 - ejemplo

```
MPI::Bcast(&dataBuffer, 1, MPI_INT, 0,
           MPI_COMM_WORLD)
```

Comunicación en MPI

- *Gather*
 - header

```
int MPI::Gather(void *sendbuf, int sendcnt, MPI_Datatype  
sendtype, void *recvbuf, int recvcnt, MPI_Datatype recvtype,  
int root, MPI_Comm comm)
```
 - ejemplo

```
MPI::Gather(&sendBuffer, 1, MPI_DOUBLE, &receiveBuffer,  
1, MPI_DOUBLE, 0, MPI_COMM_WORLD)
```
 - Debe especificar el tamaño de cada elemento de datos entrantes para el búfer de recepción, no el tamaño total del búfer

Programación en MPI

- Ejemplo: sumar un vector de 10 elementos con tres procesos
 1. Partitionar el arreglo en tamaños 4, 3, y 3
 - este debe implementarse en el mismo código para todos los procesos
 - double vector[localProblemSize]
 2. Calcular suma local
 - `for (i=0, i<localProblemSize, i++){localSum += vector[i]}`
 - en el caso de funciones globales, también se necesita conocer el primer elemento:
`localSum += cos[i + myFirstIndex]`

Programación en MPI

- Ejemplo: sumar un vector de 10 elementos en con tres procesos

- Opción 1: enviar las sumas locales al proceso maestro y sumarlas

```
if (myRank==0){  
    for(n=1, n<commSize; n++){  
        MPI_Recv(&receiveBuffer, 1, MPI_DOUBLE, n, n,  
MPI_COMM_WORLD, &status);}  
    else{  
        MPI_Send(&localSum, 1, MPI_DOUBLE, 0, myRank,  
MPI_COMM_WORLD);}  
}
```

Rango de remitente

Etiqueta de mensaje

Tamaño de arreglo local

Rango de receptor

Programación en MPI

- Ejemplo: sumar un vector de 10 elementos en con tres procesos
 - 3. Opción 2: realizar una operación de reducción
- `MPI_Reduce(&localSum, &globalSum, 1, MPI_DOUBLE,
MPI_SUM, 0, MPI_COMM_WORLD)`

Etiqueta del proceso raíz

Tamaño del arreglo local

Programación en MPI

- Operaciones colectivas de tamaño variable
 - Por ejemplo, un *gather* con diferentes tamaños de entrada en cada procesador
 - Debe especificar los tamaños de cada procesador y su ubicación en el búfer
 - **MPI::Scatterv(...)**
 - **MPI::Gatherv(...)**

Programación en MPI

- Operaciones colectivas sin bloqueo
 - Los procesadores continúan mientras esperan la comunicación
 - Se puede usar bareras para esperar el término de la comunicación
 - `MPI::Ibcast(..., MPI_Request *request)`
 - `MPI::Wait(request)`

Programación en MPI

- Comunicación de proceso a proceso sin bloqueo
 - Dos procesadores intercambian datos entre ellos
 - `MPI::Isend(..., MPI_Request *request)`
 - `MPI::Irecv(..., MPI_Request *request)`
- Se puede usar bareras para esperar el término de la comunicación
 - `MPI::WaitAll(count, array_of_requests, array_of_statuses)`

Programación en MPI

- Intercambio de datos por parejas
 - Dos procesadores intercambian datos entre ellos
 - `MPI::Sendrecv(..., rankDestiny, ..., rankSource, ...)`
- Recuerda que todos los procesos lo realizan
 - Útil para enviar datos al vecino izquierdo o derecho
 - Si un procesador, por ejemplo, el primero o último, no envía o reciba algo, se puede usar `MPI_PROC_NULL` como destino o fuente

Programación en MPI

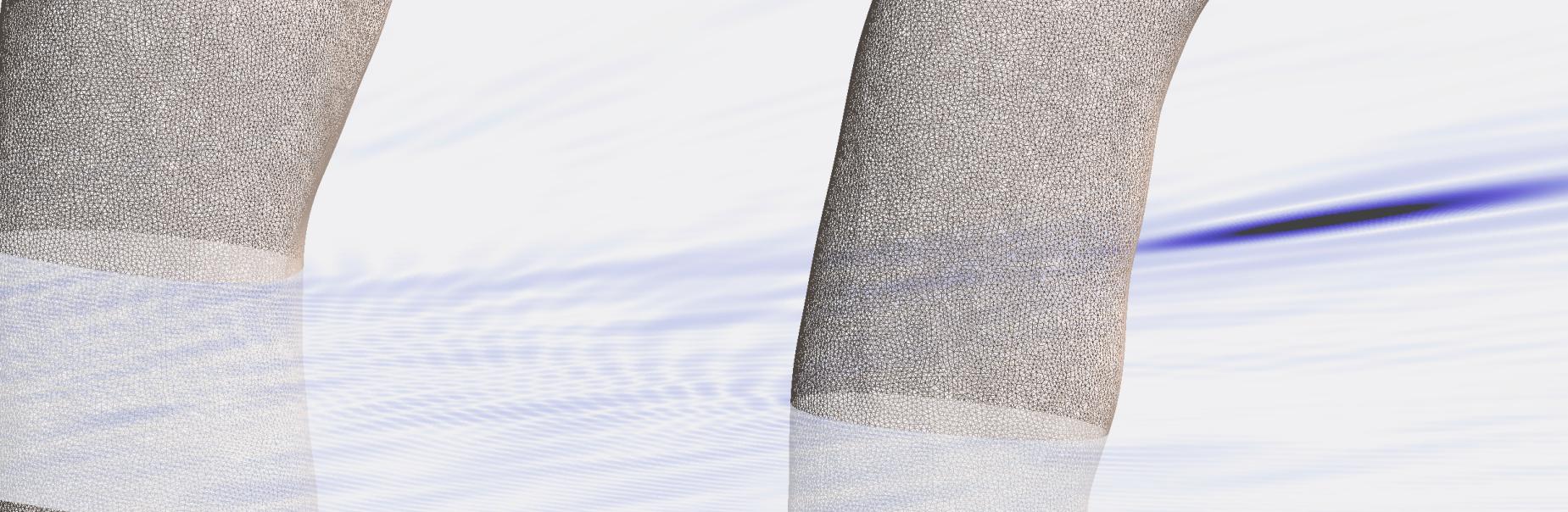
- El **MPI_COMM_WORLD** es el objeto comunicador para todos los procesos
- Se puede crear subcomunicadores
 - Útil cuando desea dividir el código en varios componentes paralelos
 - Realizar operaciones colectivos en un subcomunicador
 - Ejemplo: reducción de columnas o filas de bloques en una matriz particionada por bloques

Resumen

- Programar la comunicación en MPI

Clase siguiente

- Métodos numéricos



IMT2112 - Algoritmos Paralelos en Computación Científica

Comunicación con MPI

Elwin van 't Wout

1 de octubre de 2019



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

Facultad de Matemáticas • Escuela de Ingeniería

imc.uc.cl