DESARROLLO DE SOFTWARE EN ARQUITECTURAS PARALELAS

1. Motivación y aspectos de la programación paralela.
2. Tipos de sistemas paralelos. Paradigmas de programación paralela.
3. Conceptos básicos y medidas de paralelismo.
4. Diseño de programas paralelos.
5. La interface de paso de mensaje: el estándar MPI.
   - Programación mediante paso de mensajes.
     - Modelo de ejecución.
     - Las operaciones básicas: send/receive.
     - Operaciones de comunicación colectiva.
   - Introducción al estándar MPI (Message Passing Interface).
6. Paralelización de algoritmos: ejemplos y aplicaciones.

Modelo de ejecución

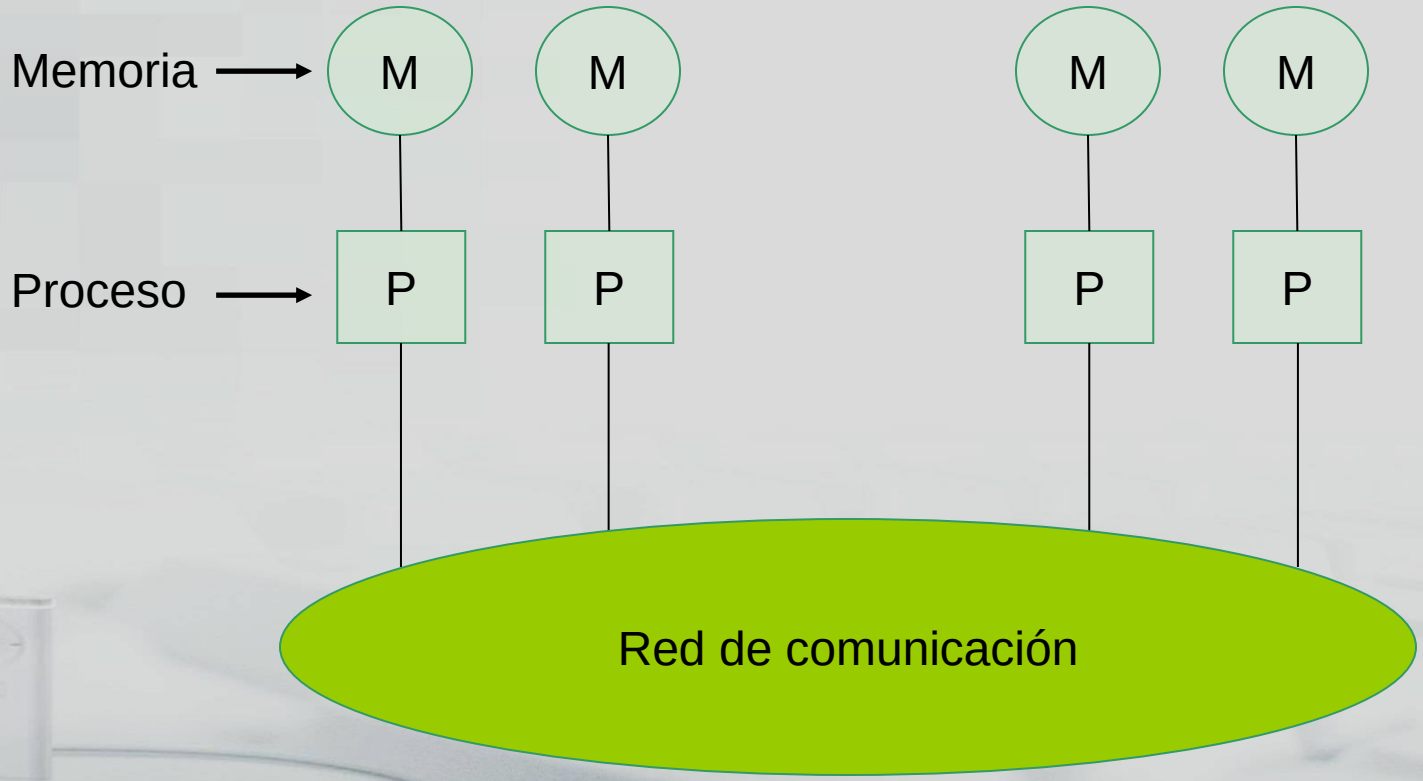
Características del modelo de paso de mensajes:

- Comprenden la utilización de una librería de subrutinas que el programador invoca de acuerdo a las necesidades del programa.
- Existe un conjunto de tareas o procesos que utilizan su propia memoria local durante la computación.
- En un mismo procesador pueden residir físicamente varios procesos pero es bastante común la ejecución de un único proceso por procesador.
- Los procesos se comunican mediante el envío y recepción de mensajes.
- La transferencia de datos entre procesos requiere operaciones de tipo cooperativo que deben ser ejecutadas en cada proceso (correspondencia entre envío y recepción).

DSAP

Modelo de ejecución

Características del modelo de paso de mensajes



DSAP

Modelo de ejecución

Estructura de un programa de paso de mensajes:

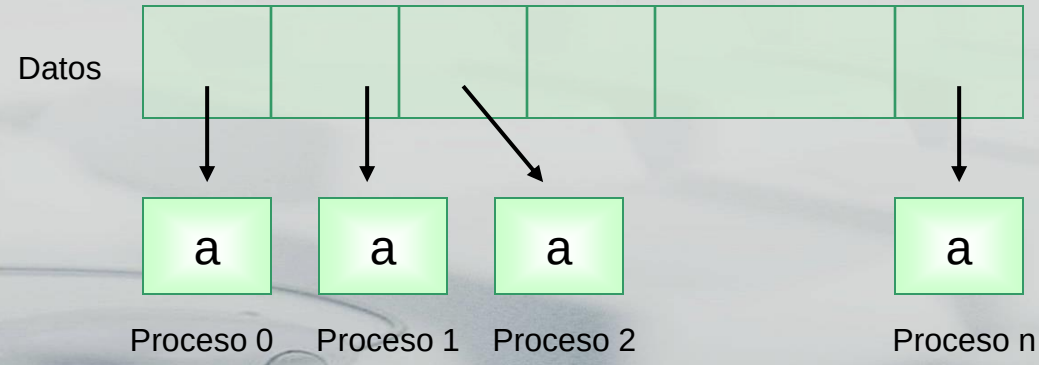
- En su forma más general, el paradigma de paso de mensajes soporta la ejecución de un programa diferente en cada uno de los procesos y el trabajo de forma completamente asíncrona en todos ellos.
  - Máxima flexibilidad.
  - Problemas: efectividad práctica, programas difíciles de entender y comportamientos no deterministas.
- En el otro extremo tendríamos el comportamiento totalmente síncrono.
  - Comportamiento, a menudo, ineficiente que le impide obtener el máximo provecho del potencial de la máquina paralela.
- La mayoría de los programas de paso de mensajes se escriben haciendo uso de un compromiso entre ambos extremos.

DSAP

Modelo de ejecución

Tipos de estrategias en la estructura de un programa de paso de mensajes:

- Programación SPMD (Single Program Multiple Data):
  - Los procesadores reciben copias idénticas del programa a ejecutar: ejecución simultánea.
  - Cada tarea se ejecuta sobre conjuntos de datos diferentes.
  - Los programas permiten que las diferentes tareas se ramifiquen (de forma condicional) dependiendo del proceso.

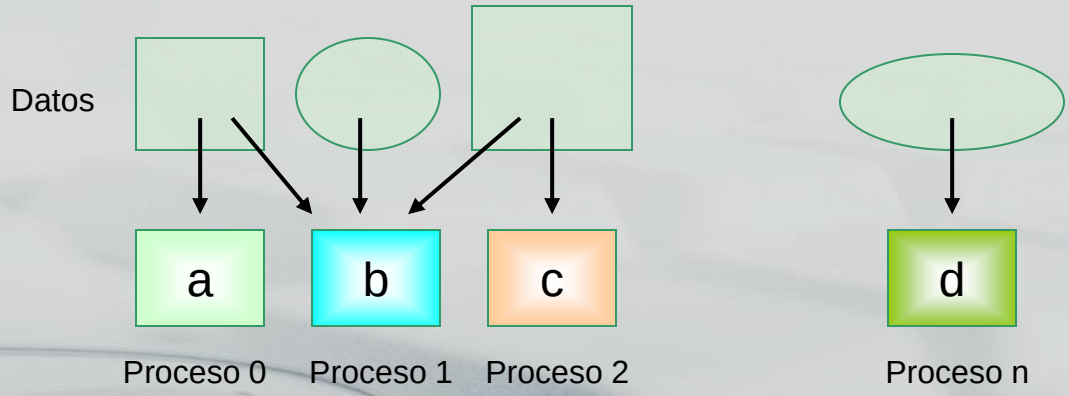


DSAP

Modelo de ejecución

Tipos de estrategias en la estructura de un programa de paso de mensajes:

- Programación MPMD (Multiple Program Multiple Data):
  - Cada tarea puede estar ejecutando el mismo o diferentes programas que otras tareas.
  - Todos los procesos pueden estar utilizando distintos conjuntos de datos.



DSAP

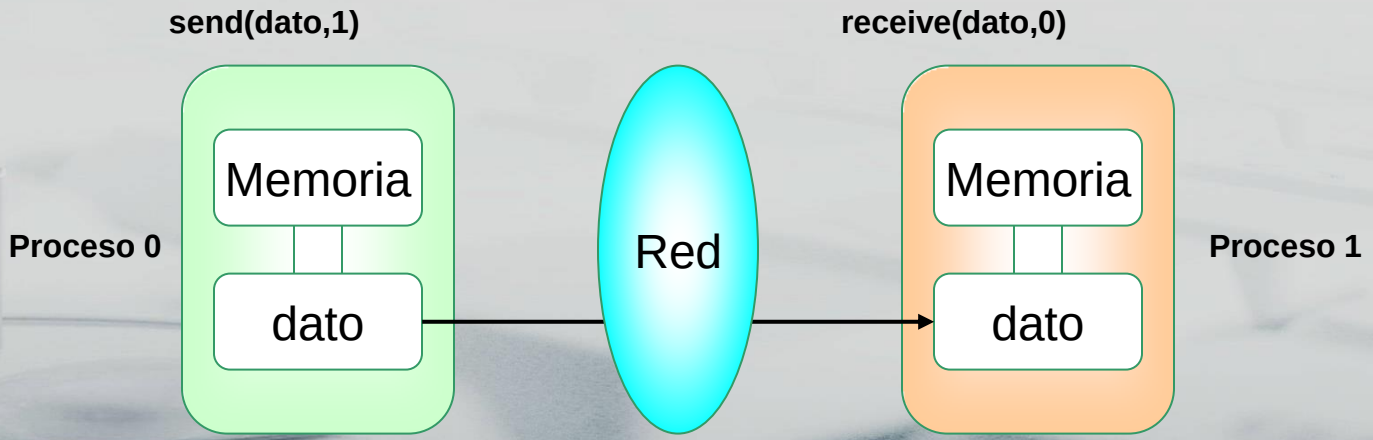
Las operaciones básicas: send/receive

El modelo de paso de mensajes se apoya en dos operaciones de comunicación básicas:

- send(Lista_de_Parametros): Envío de datos desde el proceso emisor.
- receive(Lista_de_Parametros): recepción de datos por el proceso receptor.

En el caso más simple:

- send(x, destino)
- receive(y, origen)



Las operaciones básicas: send/receive

Aspectos a tener en cuenta en el desarrollo de programas en cuanto a las operaciones send/receive:

- El sincronismo.
- El uso de buffers.
- El carácter bloqueante.

Las operaciones básicas: send/receive (sincronismo)

Operaciones síncronas/asíncronas:

- Operaciones síncronas:
  - Devuelven el control cuando ha finalizado la transferencia del mensaje.
  - El send síncrono espera a que el mensaje pueda ser aceptado por el receptor antes de iniciar el envío.
  - El receive síncrono espera a que llegue el mensaje que está esperando antes de pasar a la siguiente instrucción.
  - Por tanto, ni el proceso emisor ni el receptor podrán proceder con su ejecución hasta que el intercambio de datos finalice.
  - Garantizan una comunicación segura desde el punto de vista semántico.
- Operaciones asíncronas:
  - Los procesos pueden transferir sus datos de forma independiente, sin importar cuándo se produce realmente la recepción.
  - Permiten solapar el cómputo con la comunicación.
  - No garantizan una comunicación segura desde el punto de vista semántico.

Las operaciones básicas: send/receive (sincronismo)

Operaciones síncronas/asíncronas:

Comunicación segura y no segura desde el punto de vista semántico.

El proceso 0 envía el contenido de la variable x al proceso 1.

El proceso 1 recibe en la variable y el valor que en el proceso 0 se encuentra en la variable x.

Proceso 0

x=10

send(x, 1)
x=0

Proceso 1

receive(y, 0)

- En un send síncrono se garantiza que el valor a recibir por el receptor sea el que se encontraba almacenado en la variable del emisor antes de invocar a la operación send (en ejemplo, 10): comunicación segura.
- En un send asíncrono no se garantiza que el valor a recibir por el receptor sea 10 o 0: comunicación no segura.

DSAP

Las operaciones básicas: send/receive (sincronismo)

Interbloqueo en operaciones síncronas:

Proceso 0	Proceso 1
send(x, 1)	send(y, 0)
receive(y, 1)	receive(x, 0)

Ambos procesos pretenden intercambiar sus valores de x e y.

….pero

- La operación send (síncrona) de cada proceso está esperando el correspondiente receive del segundo proceso implicado en la operación.
- Asimismo, la operación receive de ambos procesos no se ejecuta nunca ya que la operación de envío no finaliza.

Como consecuencia, ninguno de los procesos puede proceder con su ejecución, es decir, se encuentran interbloqueados.

Las operaciones básicas: send/receive (el uso de buffers)

Operaciones con buffer/sin buffer:

Una forma de resolver el interbloqueo es mediante el uso de buffers:

- Copias temporales de los mensajes que realiza el sistema: desde el espacio de almacenamiento del usuario (definido en el proceso) hacia espacios de almacenamiento del sistema (proporcionados por la librería).
- En un send, el emisor copia el mensaje en el buffer y devuelve el control una vez realizada la copia.
- El proceso emisor puede continuar con su ejecución garantizando que cualquier cambio en los datos no tendrá impacto en la semántica.
- Los datos se copian en un buffer de recepción, de manera que cuando se ejecuta el receive se accede a él y se recoge el mensaje.

- Introducen una sobrecarga, debido a la realización de copias adicionales, pero permiten reducir los tiempos de espera en los que pueden incurrir los procesos que usen operaciones síncronas.

Las operaciones básicas: send/receive (el uso de buffers)

Interbloqueo en operaciones con buffer:

Proceso 0	Proceso 1
receive(y, 1)	receive(x, 0)
send(x, 1)	send(y, 0)

Ambos procesos pretenden intercambiar sus valores de x e y.

….pero

Como en la recepción del mensaje es habitual esperar hasta la llegada del mensaje para asegurar la semántica, ninguno de los procesos puede continuar con su ejecución porque están esperando el mensaje que nunca será enviado, es decir, se encuentran interbloqueados.

Solución: cambiar el orden de las llamadas en alguno de los procesos.

Las operaciones básicas: send/receive (op. bloqueantes)

Operaciones bloqueantes/no bloqueantes

- Una operación será bloqueante cuando no devuelve el control hasta que haya garantías de que la transferencia puede completarse con seguridad.
- Las operaciones síncronas son, evidentemente, bloqueantes.

- Una operación será no bloqueante cuando devuelven el control aún cuando no haya garantías de que la transferencia del mensaje haya sido completada con seguridad.
- Las operaciones asíncronas son, evidentemente, no bloqueantes.

DSAP

Operaciones de comunicación colectiva

- Las operaciones de tipo colectivo implican compartir datos entre más de dos procesos, que se especifican como miembros de un grupo que coordina sus operaciones.
- Las librerías de paso de mensajes suelen ofrecer rutinas para construir grupos de procesos y realizar operaciones colectivas entre sus miembros.
- Las funciones colectivas suelen ser bloqueantes.
- Podemos destacar:
  - Sincronizaciones.
  - Broadcast uno-a-todos y reducciones todos-a-uno.
  - Scatter/gather.
  - Broadcast y reducciones todos-a-todos.

Operaciones de comunicación colectiva

Sincronizaciones:

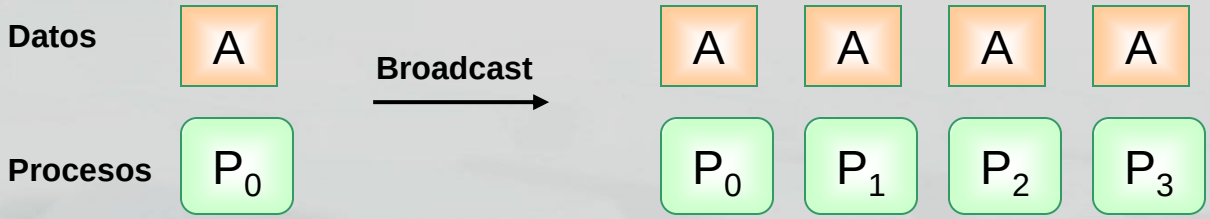
- La sincronización colectiva más habitual es la barrera o barrier.
- Están implicados todos los procesos del grupo.
- Todos los procesos deben alcanzar la barrera para poder continuar con su ejecución.

DSAP

Operaciones de comunicación colectiva

Broadcast uno-a-todos y reducciones todos-a-uno:

- El broadcast uno-a-todos envía un mismo mensaje a todos los procesos involucrados en la comunicación.
- El proceso que inicia la comunicación suele denominarse proceso raíz.
- La estrategia usada para los envíos suele estar oculta al programador a través de las rutinas de la librería que se use y, en teoría, su implementación está ligada a la arquitectura destino (válido para todas las op. colectivas).



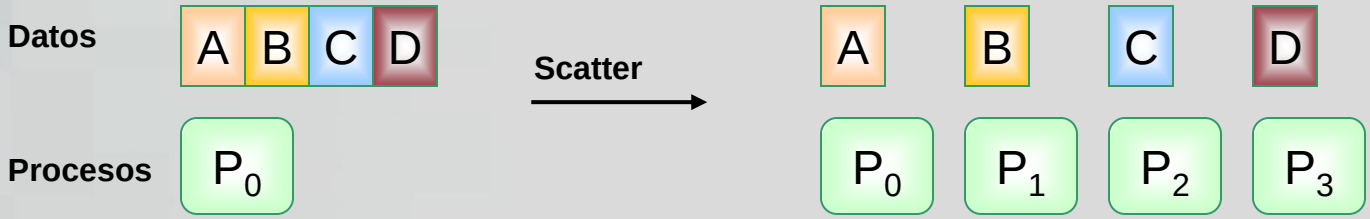
- En la reducción todos-a-uno los datos de todos los procesos se combinan mediante un operador asociativo y su resultado se acumula en un único proceso destino: suma, producto, máximo, mínimo,… de un conjunto de datos.

DSAP

Operaciones de comunicación colectiva

Scatter/gather:

- En la operación scatter, un proceso envía un mensaje individual a cada proceso del grupo.



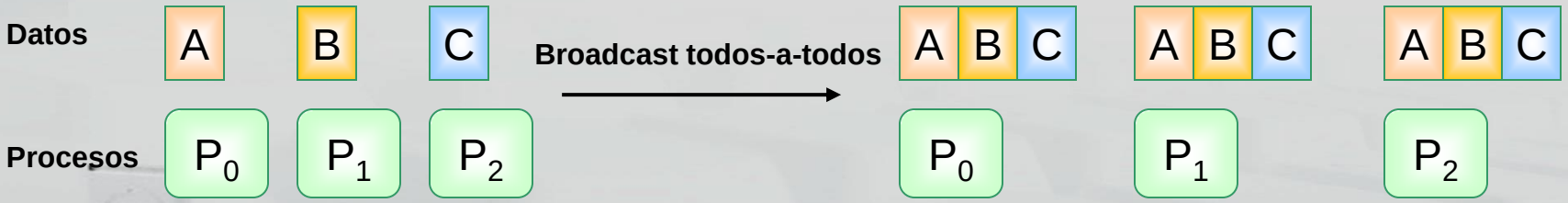
- La operación dual a la operación scatter es la operación gather (o concatenación), en la que un único proceso recoge los mensajes individuales del resto de procesos.

DSAP

Operaciones de comunicación colectiva

Broadcast y reducciones todos-a-todos:

- En el broadcast todos-a-todos, cada proceso dispone de un mensaje que debe llegar al resto de procesos en el grupo.
- Semánticamente es análogo a realizar *p* operaciones de broadcast uno-a-todos simultáneamente, una por cada proceso del grupo, en la que el proceso raíz es diferente en cada una de ellas.
- Al finalizar la operación todos los procesos deben tener almacenadas copias de los *p* mensajes.



- La operación dual es la reducción todos-a-todos en la que cada proceso es el destino de una reducción todos-a-uno.