Tema 2-Modelo de paso de mensajes

Las operaciones colectivas son aquellas en las que intervienen todos los procesos de un comunicador. Incrementan la productividad de los programas, dando mayor redimiento, reducción de errores y una codificación de más alto nivel.

MPI_Barrier

```
int MPI_Barrier(MPI_Comm comunicador);
```

Bloquea al proceso que realiza la llamada hasta que todos los miembros del grupo lo hayan llamado. La función devuelve el control al proceso solo cuando todos los procesos pertenecientes al grupo hayan realizado la llamada.

MPI_Bcast

```
int MPI_Bcast(void *dato, int tamaño_dato, MPI_Datatype tipo_MPI, int root,
MPI_Comm comunicador);
```

Envía un único mensaje desde el proceso que tenga el rango de raíz (root) a todos los procesos del grupo (en general, los del comunicador especificado), incluyendose a sí mismo. Todos los procesos llaman a la función exactamente de la misma manera, usando siempre los mismos argumentos para comm (el comunicador) y para el proceso raíz (root). Cuando se haya completado la función, todos los procesos tendrán una copia en su buffer de entrada de lo enviado por el proceso raíz.

MPI_Scatter

```
MPI_Scatter(void *dato_entrada, int tamaño_dato_entrada, MPI_Datatype
tipo_MPI_entrada, void *dato_salida, int tamaño_dato_salida, MPI_Datatype
tipo_MPI_salida, int root, MPI_Comm comunicador);
```

El funcionamiento es como si la raíz ejecutara n operaciones de envío, una a cada proceso que ejecutase la función. En cada operación se enviaría un trozo de igual tamaño del mensaje completo original de forma ordenada. Esto significa que el mensaje original se parte en n partes iguales, que es exactamente el valor que debe tener recvcount.
Los tipos tanto de envío como de recepción deben ser iguales. Esto significa que la cantidad de datos enviados debe ser la misma que el conjunto de los datos recibidos.
Todos los argumentos de la función son importantes para el proceso raíz, sin embargo

para el resto de procesos únicamente se tiene en cuenta los argumentos recvbuf, recvcount, recvtype, root. Los argumentos root y comm deben tener un valor idéntico en todos los procesos.

MPI_Gather

```
MPI_Gather(void *dato_entrada, int tamaño_dato_entrada, MPI_Datatype
tipo_MPI_entrada, int tamaño_dato_salida, MPI_Datatype tipo_MPI_salida, int
root, MPI_Comm comunicador);
```

Funciona exactamente a la inversa de MPI_Scatter.

MPI_Reduce

```
MPI_Reduce(void *dato_entrada, void *dato_salida, int tamaño_dato,
MPI_Datatype tipo_MPI, MPI_op operación, int root, MPI_Comm comunicador);
```

Realiza una operación de la siguiente lista y guarda el resultado en el buffer de salida.

Lista de operaciones MPI_Op:

- MPI_MAX: Máximo entre los elementos.
- MPI_MIN: Mínimo entre los elementos.
- MPI_SUM: Suma.
- MPI_PROD: Producto.
- MPI_LAND: AND lógico.
- MPI_ BAND: AND de bits.
- MPI_LOR: OR lógico.
- MPI_BOR: OR de bits.
- MPI_LXOR: XOR lógico.
- MPI_BXOR: XOR de bits.
- MPI_MAXLOC: Valor máximo entre los elementos y el rango del proceso que lo tenía.
- MPI_MINLOC: Valor mínimo entre los elementos y el rango del proceso que lo tenía.