Luis M Benítez Ruiz

994361

Preguntas Practicas MPI

Practica 1

No hay preguntas

Practica 2

¿Para qué sirve la instrucción MPI_Get_processor_name?

Obtiene el nombre del procesador donde se está ejecutando el proceso. Esto depende enteramente de la implementación nativa, así que puede regresar algo parecido al comando hostname, úname, etc.

Practica 3

Probar con más de dos procesos. ¿Qué sucede?

Lo mismo que con dos. Todos los procesos con rango mayor a 1 no ejecutan ninguna instrucción antes de terminar.

Practica 4

¿Cuál es la diferencia entre procesos bloqueantes y no bloqueantes?

Los procesos bloqueantes son aquellos que detienen el flujo del proceso hasta que termine su ejecución. En el caso de send y receive bloqueante, la ejecución del proceso se detiene hasta que los datos terminen de ser enviados (para send) o recibidos (para receive). Los procesos no bloqueantes no interrumpen la ejecución, pero es necesario que mantengan información de su estado para evitar problemas como intentar acceder a una variable antes de que esta exista o este inicializada completamente.

Practica 5

No hay preguntas.

Practica 6

Que hacen las siguientes funciones y cuál es su función prototipo:

MPI_Scatter:

```
int MPI_Scatter(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
               void *recvbuf, int recvcount, MPI_Datatype recvtype, int root,
               MPI_Comm comm)
```

Scatter divide el contenido de *sendbuf*, en paquetes de tamaño *sendcount* y los distribuye a los procesos que encuentre en *comm*. Los procesos receptores deben tener un buffer valido en *recvbuf*. *root* indica cual es el proceso emisor.

MPI_Gather

```
int MPI_Gather(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
               void *recvbuf, int recvcount, MPI_Datatype recvtype,
               int root, MPI_Comm comm)
```

gather reúne todos los *sendbuf*, cada uno de tamanio *sendcount* en el buffer *recvbuf* del proceso con rango indicado por *root.* Una peculiaridad es que *recvcount* debe indicar el número de elementos recibidos por cada proceso, no la cantidad de elementos a recibir en *recvbuf*. El buffer especificado en *recvbuf* solo necesita ser válido en el proceso receptor.

MPI_Allgather

```
int MPI_Allgather(const void *sendbuf, int sendcount, MPI_Datatype sendtype,
                  void *recvbuf, int recvcount, MPI_Datatype recvtype,
                  MPI_Comm comm)
```

Funciona como gather, con la diferencia de que todos los procesos participantes reciben una copia de los datos reunidos, por eso no se indica un *root*. Todos los procesos necesitan un *recvbuf* valido.

MPI_Reduce

```
int MPI_Reduce(const void *sendbuf, void *recvbuf, int count,
               MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)
```

Reduce aplica la operacion *MPI_Op* sobre cada elemento en *sendbuf* y guarda los resultados en *recvbuf*. El proceso con rango *root* es el único que require tener un *recvbuf* valido. Los parámetros *count* indican el número de elementos a reunir.

MPI_Allreduce

```
int MPI_Allreduce(const void *sendbuf, void *recvbuf, int count,
                  MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)
```

Funciona igual que reduce, con la diferencia de que todos los procesos participantes reciben una copia de los datos finales, por eso no se indica un *root*. Todos los procesos necesitan un *recvbuf* valido.

MPI_Barrier

```
int MPI_Barrier( MPI_Comm comm )
```

Bloquea cada proceso que lo llame, hasta que todos los procesos en *comm* hayan llamado a *MPI_Barrier*.

Practica 7

Pruébalo con 2 y 4 procesos. ¿Qué resultados obtienes?

Para 100,000 elementos obtengo 704982704 en ambos casos. Comprobé que fuera correcto utilizando una formula analítica. El tiempo de ejecución es el mismo para ambos casos, aunque la cantidad de datos es poca para que se pueda notar una diferencia. Adicionalmente el hilo principal realiza poco trabajo, así que su tiempo de ejecución es menor pues pasa la mayor parte del tiempo bloqueado, esperando que otros procesos realicen el trabajo.

Practica 8

No hay preguntas