Reporte Multiplicación de Matrices en MPI

Bernardo Bernal Cabrera

Méndez Gutiérrez Néstor Javier

Ingeniería en Informática

Generación: 2015

Clave UASLP:

248062

250980

Clave Fac. Ing.

201501400140

201501400847

Materia: Supercómputo,

Grupo: 222601

Fecha: 06 de octubre de 2019

Resumen

En este documento se presentan las actividades realizadas para el desarrollo de un programa que realiza la multiplicación de dos matrices (A y B) en en el lenguaje de programación C y haciendo uso de **MPI,** dichas matrices debe ser de tamaño variable, es decir que el programa también ha de validar que ambas matrices sean multiplicables.

El objetivo de la práctica es que se generen N procesos y que cada proceso calcule una porción de los elementos de la matriz resultante C.

Descripción de los datos de entrada

El programa recibe parámetros desde la terminal, estos parámetros son el número de renglones y columnas de las dos matrices, quedando el comando de ejecución de la siguiente manera**:**
mpirun -np [número de procesos] [nombre del ejecutable] [número renglones matA] [número columnas matA] [número renglones matB] [número columnas matB]

```
[root@cluster Mult_Matri_MPI]# mpirun -np 3 MultMatri 5 4 4 3
```

Descripción de los datos de salida

Se visualiza en pantalla cada una de las matrices A, B y la matriz resultante C.

Contenido

Para el desarrollo de este problema se trabajó en partes, desde lo más simple hasta lo más complejo del problema:

1. **Análisis del proceso de multiplicación de dos matrices**: en esta parte observamos a detalle el proceso de multiplicación de dos matrices desde la verificación de si un par de matrices pueden multiplicarse entre sí, hasta cómo es que cada elemento de la matriz resultante es calculado.

2. **Generar ambas matrices:** A partir de los argumentos recibidos validamos que el tamaño de las matrices sea adecuado para su multiplicación después generamos ambas matrices tal que los elementos dentro de estas fuesen aleatorios

3. **Distribución del trabajo:** Calculamos cuantos elementos de la matriz resultante debía calcular cada uno de los procesos y con la función MPI_Bcast enviamos las matrices a todos los procesos.

4. **Reserva de memoria:** En el proceso cero reservamos memoria para la matriz resultante C y también para un arreglo el cual contendrá temporalmente los datos calculados por cada uno de los procesos para posteriormente pasar los datos de este arreglo auxiliar a la matriz resultante C.

5. **Cálculo de parcial de la matriz:** En este punto se desarrolla las instrucciones que han de ejecutar los procesos diferentes del cero, dentro de cada proceso se generan tres arreglos auxiliares los cuales representan filas, columnas y un arreglo en donde se han de guardar los elementos calculados por cada proceso, una vez realizados los cálculos correspondientes a cada proceso se usa **MPI_Send** para enviar los elementos ya calculados.

6. **Unificar los elementos ya calculados:** Una vez que los procesos han terminado de calcular los elementos que se le asignaron, el proceso cero usa la función **MPI_Recv** para recibir la información de cada proceso y a su vez unificar los elementos en la memoria destinada para la matriz resultante.

Conclusión

En esta práctica se usaron las funciones **MPI_Send**, **MPI_Recv** y la recién vista **MPI_Bcast,** que una vez que hemos usado las primeras dos, usar esta tercera no nos ha conflictuado, también hemos usado el conocimiento obtenido en clase acerca de cómo paralelizar algoritmos que normalmente desarrollamos de manera secuencial, uno de los problemas con los que nos enfrentamos fue la decisión de si solo enviar la información que requiere cada proceso o enviar toda la información sin importar si parte de la información no iba a ser de utilidad para los procesos, al final decidimos enviar toda la información ya que realmente no es demasiada la información enviada, aunque estamos conscientes que es ineficiente, en el caso hipotético de que la información fuese mucha tendríamos que rediseñar el algoritmo de tal forma que cada procesos solo obtiene la información que le es de utilidad.

Avance obtenido: 100%