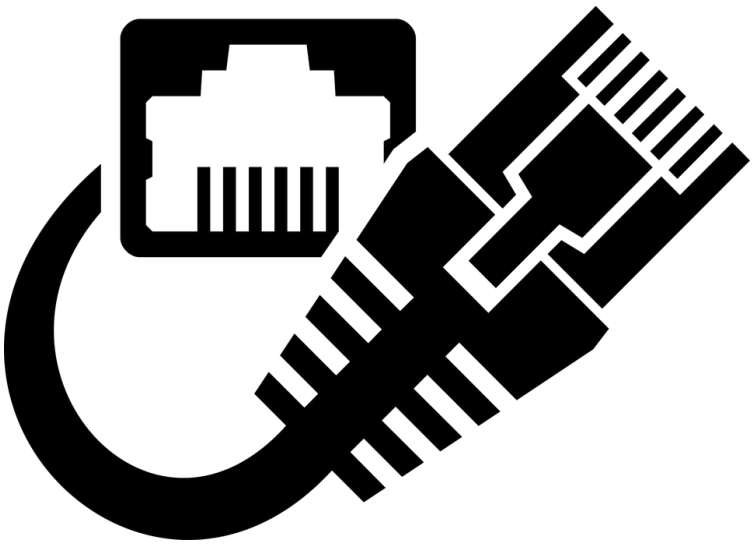
P2 - SISTEMA DISTRIBUIDO DE RENDERIZADO DE GRÁFICOS



Pablo Rodríguez Solera

Diseño de Infraestructura de Red

➔ Prof. Javier Ayllón

➔ Escuela Superior de Informática.

➔ Universidad de Castilla – La Mancha

Índice

P2 – SISTEMA DISTRIBUIDO DE RENDERIZADO DE GRÁFICOS:

1. Realización:

1.1 Enunciado:

Utilizaremos las primitivas pertinentes MPI2 como acceso paralelo a disco y gestión de procesos dinámico:

Inicialmente el usuario lanzará un solo proceso mediante mpirun -np 1 ./pract2. Con ello MPI lanza un primer proceso que será el que tiene acceso a la pantalla de gráficos pero no a disco. Él mismo será el encargado de levantar N procesos (con N definido en tiempo de compilación como una constante) que tendrán acceso a disco pero no a gráficos directamente. Los nuevos procesos lanzados se encargarán de leer de forma paralela los datos del archivo foto.dat. Después, se encargarán de ir enviando los pixels al primer elemento de proceso para que éste se encargue de representarlo en pantalla.

Usaremos la plantilla pract2.c para comenzar a desarrollar la práctica. En ella debemos completar el código que ejecuta el proceso con acceso a la ventana de gráficos (rank 0 inicial) y la de los procesos "trabajadores".

Se proporciona el archivo foto.dat. La estructura interna de este archivo es 400 filas por 400 columnas de puntos. Cada punto está formado por una tripleta de tres "unsigned char" correspondiendo al valor R,G y B de cada uno de los colores primarios. Estos valores se pueden usar para la función `dibujaPunto`.

1.2 Planteamiento de Solución:

Para llegar a la solución del problema planteado el programa realiza los siguientes pasos:

1. El proceso 0 realiza un spawn para crear tantos hijos como se le indiquen y se pone a la espera de que estos procesos envíen los puntos a dibujar.
2. Cada uno de los procesos hijo calcula que parte del fichero le corresponde leer respecto al número de hijos totales que el padre creó.
3. Cuando sabe a que parte del fichero tiene que acceder, va leyendo tripletas del mismo y se las manda al proceso padre.
4. El proceso padre va recibiendo tripletas y las dibuja sobre la pantalla formando la imagen.

1.3 Diseño del Programa:

El programa está diseñado en el lenguaje de programación c y utiliza los comandos y estructuras propios de MPI (Message Passing Interface).

El programa solo crea 1 proceso que es el padre. A partir de este, con el comando MPI_Comm_spawn se crean los demás.

El proceso padre es el encargado de dibujar, mientras que los procesos hijo son los que procesan la información del archivo, que se trata como un MPI_File y por lo tanto se usan los comandos apropiados para este, como el MPI_File_open y el MPI_File_read.

También se utiliza el comando MPI_File_set_view para posicionar a cada proceso hijo en la parte del fichero que le corresponde.

1.4 Instrucciones:

Las instrucciones de lanzamiento del programa están basadas en su compilación y ejecución siguiendo el `Makefile`.

- Para compilar el programa:

  ```
  make compilar
  ```

- Para lanzar el programa:

  ```
  make Renderizado
  ```

- Para eliminar el archivo de ejecución:

  ```
  make limpiar
  ```

2. Conclusiones

Las principal conclusión que he obtenido de la realización de esta práctica es, al igual que en la anterior, que un correcto uso de MPI nos permite realizar operaciones de computación distribuida con gran facilidad permitiendo así un gran rendimiento de estas.

3. Fuentes

Para la realización del programa se ha utilizado la documentación oficial de MPI.

https://www.mpich.org/documentation/guides/

https://www.mpich.org/documentation/manpages/