Implementación de programas paralelos usando el esquema maestro esclavo en C++ mediante la biblioteca MPI

(Antonio Carrillo Ledesma Versión 28 julio 2009)

Primeramente, hay que notar que para implementar un programa en C++ mediante la biblioteca de paso de mensajes MPI el programa básico tiene la siguiente estructura:

```
// Programa Maestro-Esclavo
int main(int argc, char *argv[])
{
  int i, j, sw, np;

  MPI::Init(argc,argv);
  MPI_id = MPI::COMM_WORLD.Get_rank();
  MPI_np = MPI::COMM_WORLD.Get_size();

   // Revisa que pueda arrancar el esquema M-E
   if (MPI_np < 2)
   {
     printf("Se necesitan al menos dos procesadores para el esquema M-E\n");
     return 1;
   }


  // Controlador del esquema M-E
  if (MPI_id == 0) // Maestro
  {
    time_t t1,t2;
    time(&t1);
    printf("\nInicio de programa\n");

    // Acciones a realizar por el nodo maestro

    printf("\nFin de programa\n");
    time(&t2);
    printf("Tiempo Cálculo: %f\n",difftime(t2,t1));
   } else { // Esclavos
    // Control de los nodos esclavos
    MPI::COMM_WORLD.Recv(&sw, 1, MPI::INT, 0,1);
    while(sw)
    {

      // acciones a realizar por los nodos esclavos

      MPI::COMM_WORLD.Recv(&sw, 1, MPI::INT, 0,1);
    }
  }

  MPI::Finalize();
  return 0;
}
```

Por lo anterior, las acciones que realice el nodo maestro y los nodos esclavos estarán desacoplados, es decir, que si se usa programación orientada a objetos, deberá separarse la jerarquía de clases para que existan al menos dos tipos de objetos, uno para el maestro y otro u otros para el esclavo, estos no podrán comunicarse directamente, para ello se usara como interfase de comunicación al paso de mensajes de MPI, donde sólo es posible enviar números (int, double, long double, etc) y arreglos de estos.

Si se parte de un programa secuencial orientado a objetos, lo ideal para evitar reescribir las clases y reusar la mayor cantidad de código, en el análisis, diseño y programación de la aplicación se deberá de tomar en consideración estas limitantes de la programación paralela mediante MPI.

Por ejemplo para un programa de descomposición de dominio, por ejemplo usando los métodos numéricos de subestructuración Schur, FETI y FETI-DP sin importar la dimensión del problema a tratar, el problema y la geometría es posible hacer la siguiente separación de clases:

- Nodo maestro: control de la parte global del método mediante peticiones a los nodos esclavos (que contendrán a los subdominios) para generar los subdominios, conocer los nodos de la frontera interior, la resolución del sistema lineal virtual asociado al método y la solución de los nodos interiores del problema.
- Nodo esclavo: mediante un arreglo de objetos que modelen a un subdominio, aceptaran las peticiones del nodo maestro, las procesaran y regresaran los resultados obtenidos al nodo maestro.

De manera esquemática las acciones a realizar podrían quedar como:

Operaciones Nodo Maestro	Operaciones Nodo Esclavo
Dado un dominio y su partición generar la malla gruesa (r=nxm número de subdominios) de la descomposición del dominio	
Dada la descomposición, dividir el número de subdominios entre el número de nodos esclavos para conocer la cantidad de objetos del tipo subdominio se generarán en cada nodo esclavo	
Solicitud a cada nodo esclavo para generar p subdominios y configurarlos con la información de la partición que le corresponda	
	Generar un arreglo de subdominios y configurarlos con la información de la partición que le corresponda
	Generar en cada subdominio la partición fina de la malla
	Enviar al nodo maestro un arreglo con la información de los nodos de frontera de cada subdominio
Una vez que se conozcan los nodos de frontera de cada subdominio, se puede calcular los nodos de la frontera interior, así como su multiplicidad (la cual variara según el esquema usado para asociar los nodos primales)	
Enviar a cada subdominio la indicación de cuales de sus nodos de frontera son duales o primales	

	Con la información de nodos de frontera interior (duales y primales) se puede calcular en cada subdominio las matrices asociadas a método numérico
Iniciar el método numérico para resolver el sistema lineal virtual asociado al método de descomposición de dominio	
Enviar el vector correspondiente con los valores de la frontera interior asociado a cada iteración del método lineal	
	Recibir el vector correspondiente con los valores de la frontera interior asociado a cada iteración del método lineal
	Hacer los cálculos de proyección para generar el nuevo vector correspondiente con los valores de la frontera interior
	Enviar el vector correspondiente al nodo maestro
Recibir todos los vectores y hacer lo necesario para continuar con la siguiente iteración del método numérico que resuelve el sistema lineal virtual	
Una vez encontrada la solución de la frontera interior, se enviara la porción correspondiente a cada subdominio	
	Recibir la solución a la frontera interior del subdominio
	Generar la solución a los nodos interiores
	Liberar los objetos de subdominio generados en cada nodo esclavo

En todas las acciones e información que el maestro envía a los nodos esclavos y que de los nodos esclavos se envían al nodo maestro, se pueden realizar mediante el paso de mensajes de números y arreglo de números. Así, si desde el análisis de la aplicación secuencial se realiza dicha separación de la jerarquía de clases, es relativamente sencillo el pasar el código secuencial a uno paralelo mediante la biblioteca de paso de mensajes MPI.

Por ejemplo si quisiéramos ejecutar en programación orientada a objetos el siguiente comportamiento:

```
sd[i].Parametros(x[0], y[0], x[2], y[2], subpart_hor, subpart_ver);
```

Este se deberá separar en dos partes, una para el maestro enviando los diferentes valores y otra en el esclavo recibiendo estos valores y ejecutando el comportamiento local al subdominio, así tenemos:

Nodo maestro:

```
arr[0] = Asig_tareas[np - 1];
arr[1] = SubParticion_H;
arr[2] = SubParticion_V;
arr[3] = ITERACIONES;
MPI::COMM_WORLD.Send(&arr, 4, MPI::INT, np, 1);
MPI::COMM_WORLD.Send(&x, 4, MPI::DOUBLE, np, 1);
MPI::COMM_WORLD.Send(&y, 4, MPI::DOUBLE, np, 1);
MPI::COMM_WORLD.Send(&TOLERANCIA, 1, MPI::DOUBLE, np, 1);
```

Nodo Esclavo:

```
MPI::COMM_WORLD.Recv(&arr, 4, MPI::INT, 0,1);
subpart_hor = arr[1];
subpart_ver = arr[2];
iteraciones = arr[3];
MPI::COMM_WORLD.Recv(&x, 4, MPI::DOUBLE, 0,1);
MPI::COMM_WORLD.Recv(&y, 4, MPI::DOUBLE, 0,1);
MPI::COMM_WORLD.Recv(&tolerancia, 1, MPI::DOUBLE, 0,1);
sd[i].Parametros(x[0], y[0], x[2], y[2], subpart_hor, subpart_ver);
```

De esta manera se deberá de desacoplar los códigos que ejecutará el nodo maestro con respecto a los que ejecutará los de los nodos esclavos.