1. Definir un entorno de programación único que garantice la portabilidad de las aplicaciones paralelas.
2. Definir totalmente el interfaz de programación, sin especificar cómo debe ser la implementación del mismo.
3. Ofrecer implementaciones de calidad, de dominio público, para favorecer la extensión del estándar.
4. Convencer a los fabricantes de computadores paralelos para que ofrezcan versiones de MPI optimizadas para sus máquinas (lo que ya han hecho fabricantes como IBM y Silicon Graphics).

Un programador que quiera emplear MPI para sus proyectos trabajará con una implementación concreta de MPI, que constará de , al menos, estos elementos

Una biblioteca de funciones para C, más el fichero de cabecera mpi.h con las definiciones de esas funciones y de una colección de constantes y macros

Comandos para compilación, típicamente mpicc, mpif77, que son versiones de los comandos de compilación habituales (cc, f77) que incorporan automáticamente las bibliotecas MPI.

Comandos para la ejecución de aplicaciones paralelas, típicamente mpirun.

**Comunicación punto a punto**

Un buen número de funciones de MPI están dedicadas a la comunicación entre pares de procesos

Existen múltiples formas distintas de intercambiar un mensaje entre dos procesos, en función del modelo y el modo de comunicación elegido.

**Modelos y modos de comunicación:**

MPI define dos modelos de comunicación: bloqueante (blocking) y no bloqueante (nonblocking). El modelo de comunicación tiene que ver con el tiempo que un proceso pasa bloqueado tras llamar a una función de comunicación, sea ésta de envío o de recepción. Una función bloqueante mantiene a un proceso bloqueado hasta que la operación solicitada finalice. Una no bloqueante supone simplemente “encargar” al sistema la realización de una operación, recuperando el control inmediatamente. El proceso tiene que preocuparse, más adelante, de averiguar si la operación ha finalizado o no.

**Comunicación básica**

El resultado de la combinación de dos modelos y cuatro modos de comunicación nos da 8 diferentes funciones de envío. Funciones de recepción sólo hay dos, una por modelo. Presentamos a continuación los prototipos de las funciones más habituales. Empezamos con MPI\_Send() y MPI\_Recv que son, respectivamente, las funciones de envío y recepción básicas bloqueantes.

**Comunicación con buffer**

Una de los problemas que tiene el envío básico es que el programador no tiene control sobre cuánto tiempo va a tardar en completarse la operación. Puede que apenas tarde, si es que el sistema se limita a hacer una copia del mensaje en un buffer, que saldrá más tarde hacia su destino; pero también puede que mantenga al proceso bloqueado un largo tiempo, esperando a que el receptor acepte el mensaje.

Para evitar el riesgo de un bloqueo no deseado se puede solicitar explícitamente que la comunicación se complete copiando el mensaje en un buffer, que tendrá que asignar al efecto el propio proceso. Así, se elimina el riesgo de bloqueo mientras se espera a que el interlocutor esté preparado. La función correspondiente es MPI\_Bsend(), que tiene los mismos argumentos que MPI\_Send().

**Recepción por encuesta:**

Las funciones de recepción de mensajes engloban en una operación la sincronización con el emisor (esperar a que haya un mensaje disponible) con la de comunicación (copiar ese mensaje). A veces, sin embargo, conviene separar ambos conceptos. Por ejemplo, podemos estar a la espera de mensajes de tres clases, cada una asociada a un tipo de datos diferente, y la clase nos viene dada por el valor de la etiqueta. Por lo tanto, nos gustaría saber el valor de la etiqueta antes de leer el mensaje. También puede ocurrir que nos llegue un mensaje de longitud desconocida, y resulte necesario averiguar primero el tamaño para así asignar dinámicamente el espacio de memoria requerido por el mensaje. Las funciones MPI\_Probe() y MPI\_Iprobe() nos permiten saber si tenemos un mensaje recibido y listo para leer, pero sin leerlo. A partir de la información de estado obtenida con cualquiera de estas “sondas”, podemos averiguar la identidad del emisor del mensaje, la etiqueta del mismo y su longitud. Una vez hecho esto, podemos proceder a la lectura real del mensaje con la correspondiente llamada a MPI\_Recv() o MPI\_Irecv()