Extendiendo el modelo de Programación de MPI

J.Fernandez, M. Fuentes, D. Funez, M. Printista, F. Piccoli *
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950
5700 - San Luis
Argentina
e-mail: {jmfer, gfuentes, dgfunez, mprinti, mpiccoli}@unsl.edu.ar

1 Introducción

En el Modelo de programación definido por la librería *MPI*[4] [6] [8], una computación comprende uno o más procesos, los cuales se comunican a través de mensajes. Los procesos envían y reciben mensajes entre sí. En al mayoría de las implementaciones *MPI*, un número fijo de procesos es creado al inicio del programa. Cada proceso puede ejecutar distintos programas, de aquí que muchas veces se referencia al modelo de programación como *MIMD* (*Múltiples Programas Múltiples Datos*).

MPI permite a los procesos realizar distintas operaciones a través del pasaje de mensajes[7] [9], entre ellas están:

- Comunicaciones *punto-a-punto*.
- Formar grupos de procesos.
- Realizar operaciones globales entre procesos de un mismo grupo, tal como las comunicaciones colectivas y las operaciones de sincronización.

Una de las características principales de *MPI* es que soporta la programación modular. Esto es posible a través de un mecanismo llamado *comunicador*, el cual define un módulo capáz de encapsular las comunicaciones internas a él. Al inicio

*Grupo subvencionado por la UNSL y ANPCYT (Agencia Nacional para la Promoción de la Ciencia y Tecnología)

de toda computación, todos los procesos pertenecen a un mismo comunicador, el comunicador global.

A pesar de las características del modelo de programación propuesto, existen aplicaciones para las cuales las herramientas provistas no son suficientes. Entre las características no soportadas por *MPI* se encuentran los mecanismos necesarios para proveer:

- Comunicación entre Threads
- Acceso Directo a Memoria Remota

En la próxima sección explicamos cada uno de estos aspectos, las situaciones en las que son necesarios y las necesidades básicas para su implementación.

2 Características de las Extensiones

Cada una de las extensiones al modelo se constituye en una libreria anexa a *MPI*. Cada librería se caracterizará por tener una interface similar a la provista por *MPI* y, la comunicación se efectivizará mediante la transmisión de mensajes entre procesos. En otras palabras estas extensiones definen una capa superior, siendo *MPI* una capa inferior a ésta. La figura 1 muestra un esquema del sistema.

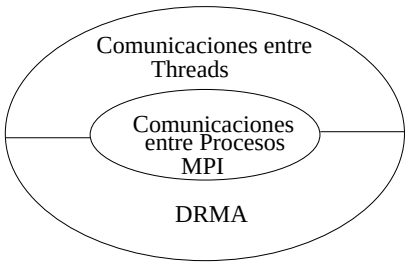

Figura 1: Estructura del nuevo Sistema de Comunicaciones

A continuación detallamos cada una de las extensiones.

2.1 Comunicación entre Threads

Una computación paralela ideal implica tener un procesador por proceso paralelo y la misma carga de trabajo para todos los procesadores. Esto, en la mayoría de los casos, no se cumple. Existen situaciones donde varios procesos se ejecutan sobre un único procesador. Una de ellas es la derivada de las tareas relacionadas con el *Balance de la Carga* del procesador: múltiples tareas son asignadas a un procesador con el objeto de balancear la carga[1]. La asignación se hace según el trabajo que implica cada tarea. Otra situación donde pueden existir varios procesos por un

procesador, es aquella derivada de la división del trabajo de una tarea en diferentes subtareas, las cuales superan en número la cantidad de procesadores disponibles.

La creación de procesos independientes permite mayor flexibilidad en los programas, aunque no en todos los casos es la solución adecuada. Cuando las acciones de cargar un programa a memoria, ubicar un nuevo proceso para él y, en forma regular, hacer el switching entre los distintos procesos, reperesentan una carga extra en el sistema e insumen demasiado tiempo, es bueno pensar en otras alternativas de trabajo. Una de esas alternativas es reemplazar a los procesos comúnes por los procesos de *bajo peso* o *threads*[3].

Los *threads* pertenecen a un proceso y tienen, además de la ventaja de poder comunicarse a través de la memoria del proceso, un bajo costo asociado a su *switching*. El switching de threads es menos costoso que el de procesos.

Considerando las ventajas de los threads respecto a los procesos, podemos pensar en reemplazar los procesos por threads. Ahora surge la siguiente inquietud ¿Cómo se comunican los threads que pertenecen a distintos procesos?.

Con el objetivo de proveer comunicación entre threads, pertenecientes o no a un mismo proceso, se extiende el modelo de programación de MPI incorporando una nueva librería con funciones para el pasaje de mensajes entre threads[5]. Estas comunicaciones pueden establecerse, tanto entre threads de un mismo proceso, como entre threads de distintos procesos, e incluso ejecutando en distinto procesador.

Las funciones para comunicar threads comienzan con la sigla $MPIT\_XXX$ y se distinguen de las funciones MPI_XXX en que el origen y el destino de la comunicación son, ahora, identificadores de threads. Esto implica la existencia de *comunicadores* de threads.

Un *comunicador* de threads está definido por:

- Un comunicador de procesos.
- Información relacionada a los threads: Threads que pertenecen al comunicador, Identificador del Proceso al que pertenece cada thread, etc.
- Información relacionada a las comunicaciones pendientes: Envíos y Recepciones.

La librería MPI_Th provee las funciones para:

- Creación y Eliminación dinámica de threads del sistema de Comuniación.
- Comunicaciones *Punto-a-Punto*.
- Comunicaciones Globales tipo *Broadcast*
- Operaciones de sincronización por barrera

Para proveer todas estas funcionalidades es necesario contar con un sistema para la administración de los threads, involucrados en las comunicaciones o que pueden participar de ellas, y las comunicaciones. El objetivo principal del sistema es efectivizar las comunicaciones entre los threads considerando que se realizan a nivel de procesos. Esto hace que la administración del sistema de nombres para threads o $rank\_t$ sea fundamental.

3 Acceso Directo a Memoria Remota

El acceso directo a memoria remota, $DRMA$, es otra manera de comunicación entre procesos. Este tipo de comunicaciones es también denominadas *Comunicaciones One-Sided*[2]. Un proceso puede acceder y/o modificar el contenido de un área de memoria perteneciente a otro proceso. Dicha área debe haber sido habilitada para tal fin por el proceso correspondiente.

La libreria *MPI_DRMA* debe incluir las funciones necesarias para proveer $DRMA$. Las funciones deberán permitir:

- Habilitar y Deshabilitar una región de memoria para permitir o finalizar el accedo remoto directo.
- Leer directamente el contenido de una región de memoria remota.
- Escribir directamente el resultado de una expresión en una región de memoria remota.

Para proveer todas estas funcionalidades es necesario proveerdisponer un sistema que administre las regiones de memoria pertenecientes a un proceso y que pueden ser accedidas directamente desde procesos remotos. El objetivo principal del sistema es efectivizar los accesos directos a memoria, administrando la información necesaria para tal fin.

Al igual que las comunicaciones *punto-a-punto*, la comunicaciones *one-sided* se deberan realizar entre procesos que pertenecen a un mismo comunicador.

4 Conclusiones

La extensión de la librería MPI nos define un nuevo modelo de programación, el cual puede ser aplicado para resolver aplicaciones paralelas de distinta naturaleza. Además, con la incorporación de estos dos tipos de comunicaciones, se extiende la posibilidad de aplicar algunos aspectos de los modelos de Computación como BSP en el diseño de algoritmos.

Actualmente se está trabajando en la librería MPI_Th. Algunos resultados preliminares muestran una interesante mejora respecto a MPI.

Referencias

[1] Ayguade E., Martorell X., Labarta J., Gonzalez M. and Navarro N. Exploiting Multiple Levels of Parallelism in OpenMP: A Case Study Proc. of the 1999 International Conference on Parallel Processing, Aizu (Japan), September 1999.

[2] Bonorden O., Huppelshauser N., Juurlink B., Rieping I. PUB library, Release 6.0 - User guide and function reference. University of Paderbon, Germany. (1998)

[3] Kleiman, S. Shah, D. Smaalders, B - Programming with Threads. Prentice Hall. 1996.

[4] MPI Forum: MPI-2: Extensions to the Message-Passing Interface, http://www.mpi-forum.org/docs/mpi-20.ps.Z (1997).

[5] Nitcvhe, T. - Thread Communication over MPI. Recent Advances in Parallel Virtual Machine and Message Passing Interface. Lectures Notes in Computer Science. Pp 145-151. Hungary. 2000.

[6] Pacheco, P. - Parallel Programming with Mpi - University of San Francisco - Morgan Kaufmann Publishers, Inc Copyright 1997

[7] Quinn M. Parallel Computing. Theory and Practice. Second Edition. McGraw-Hill, Inc.

[8] Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J. MPI: The complete Reference. Cambridge, MA: MIT Press, 1996.

[9] Wilkinson, B., Allen M. - Parallel programming: Techniques and Application using Networked Workstations, Prentice-Hall. 1999.