Mecánica Computacional Volumen XV
Compilado por A. Larreteguy y M. Vénere
Bariloche, noviembre de 1995

PROCESAMIENTO DISTRIBUIDO EN UNA RED DE MICROPROCESADORES: UN SISTEMA DE COMUNICACIONES

Victorio E. Sonzogni
Grupo de Tecnología Mecánica - INTEC
CONICET - Universidad Nacional del Litoral
Güemes 3450, 3000 Santa Fe, Argentina

RESUMEN

En este trabajo se ha desarrollado un sistema para comunicación de microprocesadores de manera que éstos puedan ser utilizados como nodos de cálculo de una máquina paralela virtual. Se requiere una red de computadoras personales. El presente desarrollo se realizó para DOS como sistema operativo, pero su alcance es más general. El sistema de comunicación consiste en un paquete de subrutinas FORTRAN que pueden ser llamadas desde un programa de aplicación cualquiera. El modelo de programación paralela adecuado para este entorno es el de *intercambio de mensajes*, es decir que cada procesador (nodo) ejecuta su propio programa sobre sus propios datos. Estos pueden corresponder a un subdominio, como en el caso de métodos de descomposición de dominio. El algoritmo general debe prever las comunicaciones necesarias a fin de tratar el problema global, o sea resolver el problema sobre la interfase. Se describe el sistema de comunicaciones y se presenta un ejemplo de su utilización.

ABSTRACT

A software system to communicate microprocessors, in order to use them as nodes of a virtual parallel machine, have been developed. A network of personal computers is required. The present development was carried out under DOS as operating system, but its scope is more general. The system consists in a package of FORTRAN routines which may be called from any application program. A *message-passing* programming model is suitable for this architecture, i.e. each node run its own code with its own data. These data may belong to a part of the problem, as in the case of a subdomain decomposition, for instance. The general algorithm should take care of the necessary communications. The communication system is presented, as well as an example of its use.

INTRODUCCION

El desarrollo tecnológico de las computadoras, tendiente a aumentar la potencia de cálculo, condujo a la utilización de procesamiento en paralelo. Paralelamente a estas innovaciones en la construcción de computadoras, se ha producido un desarrollo de herramientas de software para poder utilizar una red de computadoras como una única máquina paralela. Este tipo de arquitectura (redes heterogéneas de computadoras) resulta muy atractivo debido a su economía, versatilidad y rendimientos que pueden obtenerse.

Existen varios sistemas de comunicación para computadoras distribuidas. Los hay desde software específico de computadoras paralelas (NX2 de INTEL iPSC/860, o CS-TOOLS de Meiko), hasta sistemas más o menos portables como PVM, EXPRESS, LINDA, etc.
PVM (Parallel Virtual Machine) es un paquete de software, desarrollado en el Oak Ridge National Laboratory, que ha adquirido popularidad y resulta un sistema bastante versátil y portable [1]. Sin embargo corre únicamente bajo Unix. Esto puede resultar una limitación si se dispone, como en INTEC, de una cantidad de microprocesadores con otros sistemas operativos. Por este motivo se ha desarrollado un sistema que realiza -parcialmente- un trabajo como el de PVM. Este sistema se ha denominado PCCOM y se ha desarrollado en DOS. Sin embargo no existe, a priori, impedimento para su funcionamiento bajo cualquier otro sistema operativo.
El modelo natural de programación para este tipo de arquitectura es el de *intercambio de mensajes*. En general, cuanto mayor es la granularidad de tareas paralelas (i.e. la cantidad de operaciones en relación a la cantidad de comunicaciones) tanto mayor será la eficiencia paralela.
A continuación se realiza una breve presentación del sistema PCCOM, con algunos comentarios acerca de los requerimientos de hardware/software y de los programas de aplicación. Se describen brevemente las rutinas del sistema y finalmente se muestra un ejemplo sencillo de utilización del mismo.

REQUERIMIENTOS DEL SISTEMA

El sistema PCCOM se desarrolló sobre una red de computadoras personales, con procesadores INTEL 486 (o bien 386, o Pentium). La red debe funcionar sobre un soporte físico adecuado (cable Ethernet, o fibra óptica) y un software que lo maneje. En este caso se utilizó NetWare Lite, de Novell, para poder acceder discos de cualquiera de las máquinas componentes, pero cualquier otro software similar puede ser usado.
La comunicación, así como la sincronización, se efectúa a partir de envios de mensajes entre los procesadores. Y ésto se maneja, desde los programas de aplicación, mediante construcciones FORTRAN. Por lo tanto, aún cuando este trabajo se realizó bajo DOS, no debiera haber restricciones en lo que hace al sistema operativo.

PROGRAMAS DE APLICACION

Para realizar un cálculo en paralelo se precisan dos tipos de programas. Uno de ellos es un *programa madre* (*host*) que corre sobre un procesador cualquiera de la red, que se denominará *procesador madre*. El otro tipo de programa es un *programa nodo*, que corre sobre cada uno de los restante procesadores de la red. La cantidad de nodos corresponde a la cantidad de tareas que pueden ejecutarse en paralelo.
Cualquiera de los procesadores puede actuar como *madre*. El programa *madre* se encarga de montar la "máquina paralela virtual". Convoca a la cantidad de nodos necesarios y les asigna un identificador de proceso. Una vez que el sistema paralelo está montado, el proceso madre da la orden de inicio de tareas a cada nodo. El proceso madre puede (aunque no es necesario) realizar parte de las tareas paralelas, o sea calcular como un nodo más. Finalmente el proceso madre recibe los avisos de tarea cumplida de cada nodo.
El programa *nodo* se ejecuta en cada uno de los restantes procesadores. Es el que realiza la tareas en paralelo. Precisa que previamente el programa madre: 1) lo haya convocado; 2) le haya asignado un identificador de proceso; y 3) le haya dado la orden de iniciar el procesamiento paralelo. Los nodos se comunican entre sí y con el programa madre.
Una vez compilados, la construcción del módulo ejecutable para ambos tipos de programa debe ser realizada con auxilio de una librería de módulos objetos de PCCOM, y deben incluir, con la sentencia INCLUDE de FORTRAN, el archivo `COM486.H`.
La simplicidad del sistema hace que un programa nodo no sea ejecutado a partir de instrucciones FORTRAN desde otro programa. Por lo tanto se requiere que el usuario inicie la ejecución del programa nodo en cada uno de los procesadores deseados. Estos programas quedan a la espera de la convocatoria por parte del programa madre. Hay dos posibilidades para configurar la máquina virtual. Una de ellas es la definición explícita de la configuración deseada, por medio

del archivo CONFIG.H. Allí se debe indicar el número de procesos NUMPRO, y el identificador de la CPU para cada componente. La otra posibilidad es la configuración automática. El usuario indica, en el archivo CONFIG.H, solamente el número de procesos deseados NUMPRO. El proceso madre llama entonces a todos los procesadores conocidos en la red y espera su respuesta. Los primeros procesadores que responden, en el número de procesos deseado, son efectivamente convocados para conformar la máquina virtual y reciben su correspondiente identificador de proceso.

SUBRUTINAS BASICAS DEL SISTEMA

El paquete PCCOM consiste en un sistema de subrutinas y funciones FORTRAN que pueden se llamadas desde un programa de aplicación cualquiera. Algunas de estas subrutinas adquieren forma diferente según sean utilizadas por un programa madre o por un programa nodo. La versión utilizable desde el programa madre se identifica con una letra H en su nombre (de *Host*), mientras que la versión utilizable desde un nodo, se identifica con una letra N (de *Node*). A continuación se describen brevemente las principales subrutinas PCCOM. Una descripción más detallada de las mismas se brinda en referencia 2.

Configuración de la máquina virtual

La primera subrutina a ser llamada debe ser PCHOPEN en el programa madre, o PCNOPEN en el programa nodo. Esta rutinas establecen la comunicación entre los procesos. Ambas, a su vez, utilizan otras subrutinas del paquete PCCOM, especialmente aquellas que envían y reciben mensajes.

- PCHOPEN comienza leyendo del archivo CONFIG.H la cantidad de procesos requeridos NUMPRO y, opcionalmente, la lista de microprocesadores participantes. Estos últimos son convocados para constituir la máquina paralela virtual. Si no se provee esa lista, es decir que solamente se especifica la cantidad de procesos deseados, entonces el programa madre lanza un llamado a todos los microprocesadores conocidos en la red y espera la respuesta. Los primeros NUMPRO procesadores en responder constituirán la máquina virtual.
- PCNOPEN es la contraparte de PCHOPEN en el programa nodo. Esta rutina comienza un ciclo de espera de la convocatoria del proceso madre. Si la recibe, contesta comunicándole su identificador de CPU. Luego espera que la *madre* le asigne un número de proceso. A partir de allí comienza a desarrollar las tareas en paralelo. Pasado un tiempo de espera, si no tiene respuesta del programa madre, el programa nodo termina su ejecución.

Cada microprocesador de la red posee su propio *identificador de CPU*, que está escrito en un archivo denominado IDENPC.H en su propio disco. A su vez, a cada proceso el programa madre asigna un *identificador de proceso*, que es un número entre 1 y NUMPRO.

Subrutinas de comunicación

Como ha sido indicado, se utilizan diferentes rutinas para los programas madre y nodo. A su vez las rutinas de comunicación se agruparán en aquellas para *enviar* mensajes y aquellas para *recibirlos*.

ENVIO DE MENSAJES

El programa *madre* puede enviar mensajes a los nodos por medio de: PCHSEND, PCHSENDW y PCHBRDCT.

- PCHSEND envía un mensaje a un nodo especificado. El mensaje se coloca en un arreglo FORTRAN y posee un *identificador de mensaje*. El envío se realiza escribiendo en un archivo que reside en el disco del nodo receptor. Este envío es de tipo *no bloqueante*. Esto significa que el programa madre continua su ejecución luego de mandar el mensaje, sin esperar que el mismo haya sido leido.
- PCHSENDW es similar a la anterior pero *bloqueante*, por cuanto el programa madre espera a que el nodo haya recibido el mensaje, antes de proseguir su ejecución.

- PCHBRDCT envía un mensaje a *todos* los nodos (*broadcasting*).
El programa *nodo* puede enviar mensajes con las rutinas: PCNSENH, PCNSENN, PCNSENHW y PCNSENNW.
- PCNSENH envía un mensaje al programa madre, mientras que PCNSENN lo hace a otro nodo. Ambas rutinas son de carácter *no bloqueante*. El mensaje se escribe siempre en el disco del procesador receptor.
- Análogamente PCNSENHW y PCNSENNW realizan envíos de tipo *bloqueante*.

RECEPCION DE MENSAJES

Una única subrutina se utiliza para recibir mensajes en el programa madre: PCHRECV. También una subrutina se utiliza en el programa nodo para ese fin: PCNRECV. En ambos casos el programa receptor busca un archivo correo en su propio disco y luego de leer el mensaje procede a borrar ese archivo. La eliminación del archivo portador del mensaje se utiliza como "aviso de retorno" para el proceso que lo envió. En ambas rutinas se especifica el *identificador del proceso remitente* y el *identificador del mensaje* esperado. Solamente se leen los mensajes para los que ambos identificadores, contenidos en el mensaje, coinciden con los esperados por el proceso receptor. Normalmente un valor -1 en estos identificadores hace que el receptor acepte cualquier valor de identificador.

Subrutinas de empaquetado/desempaquetado

Como la comunicación es una tarea cara en el procesamiento, puede ser interesante concentrar la información a enviar de modo de reducir el número de envíos. Variables FORTRAN de diferentes tipos (por ej, REAL, INTEGER, CHARACTER) pueden ser empaquetadas juntas en un mismo mensaje. Esto se hace por medio de las subrutinas PCINITS, para inicializar el empaquetado, y PCKI2, PCKI4, PCKR4, PCKR8 o PCKS, para empaquetar variables enteras o reales, o un conjunto de caracteres.
Inversamente, luego de recibido un mensaje su contenido puede desempaquetarse utilizando las subrutinas: PCUNKI2, PCUNKI4, PCUNKR4, PCUNKR8 o PCUNKS.
Es responsabilidad exclusiva del programador que las variables sean desempaquetadas de la misma forma en que fueron empaquetadas en el programa remitente.

Subrutinas de registro de tiempo

Algunas rutinas pueden utilizarse para registrar el tiempo o para esperar un intervalo especificado. Estas son: PCINIRLJ, PCRELOJ, PCRELOJP, PCESPERA, PCWAIT, PCTIMACC y PCGETTIM.
Se utilizan dos origenes para medir el tiempo en el programa: un origen *total* y otro *parcial*. El origen *total* se pone a cero una vez al comienzo del programa con la subrutina PCINIRLJ. El origen de tiempo parcial es colocado a cero en cada llamada a la subrutina PCRELOJ.
- PCINIRLJ: inicializa el contador de tiempo total. Esta subrutina es llamada por PCHOPEN y por PCNOPEN. Normalmente el usuario no debe invocarla.
- PCRELOJ: es un cronómetro, mide el tiempo y da algunas medidas del mismo. Calcula tres medidas de tiempo: el tiempo total transcurrido desde el inicio; el tiempo parcial, desde la llamada previa a esta subrutina; y el valor acumulado de los registros parciales. Esta subrutina pone a cero el contador de tiempo parcial.
- PCRELOJP: es similar a la rutina PCRELOJ pero no pone a cero el contador de tiempo parcial. Por este motivo la subrutina PCRELOJP puede ser llamada dentro de un ciclo iterativo y en cada caso calculará el tiempo parcial con respecto a un único origen de tiempo parcial establecido previamente fuera del ciclo iterativo.
- PCESPERA: acumula, en una variable real, el tiempo parcial transcurrido. Hace un llamado interno a PCRELOJP y por tanto no modifica el origen de tiempo parcial.
- PCTIMACC: acumula, en una variable real, el tiempo parcial transcurrido. Hace un llamado interno a PCRELOJ y por tanto pone a cero el contador de tiempo parcial.

- PCWAIT: hace que el programa entre en un intervalo de espera, durante un módulo arbitrario de tiempo.
- PCGETTIM: devuelve el tiempo acumulado que el proceso ha pasado dentro de las principales subrutinas del paquete PCCOM.
Las medidas de tiempo se realizan por medio de la subrutina GETTIM de FORTRAN, con una precisión de centésimas de segundo.

Subrutinas de sincronización

La subrutina PCBARR coloca una barrera. Se define en un parámetro de la misma el número de procesos que deben darse cita en la barrera (usualmente el número total de procesos). Cuando un proceso alcanza la barrera debe esperar allí hasta que el número indicado de procesos haya arribado. Solamente cuando todos han llegado, pueden continuar la ejecución.
Esta subrutina puede ser llamada indistintamente por el programa madre o por los nodos. El contador de la barrera se almacena en un archivo en el disco del proceso madre.

Subrutinas de información

Algunas subrutinas permiten requerir el sistema PCCOM informaciones, tales como:
- PCMYPID: devuelve el identificador de proceso propio.
- PCHOSTID: devuelve el identificador de proceso madre.
- PCNUMPRO: devuelve el número total de tareas paralelas.
- PCGETINT: devuelve el valor actual de algunos parámetros enteros de de PCCOM, tales como: la unidad de salida para impresión de mensajes; el código opción de impresión de mensajes y errores; y el número de repeticiones de un módulo arbitrario de tiempo durante la espera.
- PCGETFLT: devuelve el valor actual de algunos parámetros reales de PCCOM, tales como el tiempo máximo especificado para esperar una conexión.
- PCPERROR: imprime mensajes de error.

Subrutinas de asignación de valores

Algunos parámetros pueden ser cambiados durante la ejecución en un nodo determinado. Esto se puede realizar con las subrutinas:
- PCSETINT: asigna nuevo valor a algunos parámetros enteros de PCCOM, tales como: la unidad de salida para impresión de mensajes; el código opción de impresión de mensajes y errores; y el número de repeticiones de un módulo arbitrario de tiempo durante la espera.
- PCSETFLT: asigna un nuevo valor a algunos parámetros reales de PCCOM, tales como el tiempo máximo especificado para esperar una conexión.

EJEMPLO DE UTILIZACION

En anexo se muestra un program sencillo que ilustra el uso del sistema PCCOM. Este programa ha sido tomado de un ejemplo de uso de PVM [1], ya que es simple y además permite comparar el uso de ambos sistemas de comunicación.
Ese programa utiliza un modelo de programa *master/slave*. Hay dos programas diferentes. Uno de ellos es un programa *madre* que inicia el trabajo convocando a los nodos de cálculo. Eventualmente les envía datos y recibe resultados. El otro tipo es un programa *nodo* que realiza efectivamente las tareas en parelelo.

AGRADECIMIENTOS

Este trabajo se realiza con apoyo del CONICET (PID 238), de la Universidad Nacional del Litoral (CAI+D 94/95: Métodos Numéricos en Mecánica de Sólidos y Fluidos), y de Fundación Antorchas (Proyecto 13015/1-16).

REFERENCIAS

1. **Geist, A., et al.**, *PVM 3 User's Guide and Reference Manual, Rep.*, ORNL/TM-12187, Oak Ridge National Laboratory, Oak Ridge, Tennessee USA, 1993.
2. **Sonzogni, V.E.**, *PCCOM: A Software System to Perform Concurrent Computations with a Message Passing Programming Model under DOS*, Internal Report GTM94-5, INTEC, 1994.

Anexo
Ejemplo de un programa FORTRAN

```
      program master1
c ********************************************************
c Example of fortran program illustrating the use of PCCOM
c "Host" or "Master" program
c ********************************************************
      integer idfrom,msgtyp,len,ierr
      integer lenr, numpro
      real*8  buf(10), result(32), data(100)
c
      include 'com486.h'
c     ==================
c
c ------- Starting up all the tasks ---------------
c     Enroll this program in PCCOM
      ierr = 0
      open(99,file='mst.out')
      iopt = 2
c
c  this is the first mandatory routine of PCCOM:
      call pchopen (ierr)
c     ============
      if (ierr.ne.0) go to 900
      call pcnumpro (numpro)
c     =============
c
c ------ Begin user program ----------------------
c
      n = 10
c     Initiate data array
      do 20 i=1,n
        data(i) = 1
 20   continue
c     broadcast data to all node programs
      msgtyp = 1
      call pcinits (buf,len)
c     ============
      call pcki4 (n, 1, 1, buf, len, ierr)
c     ==========
      call pckr8 (data, n, 1, buf, len, ierr)
c     ==========
      call pchbrdct (msgtyp, buf, len, ierr)
c     =============
      if (ierr.ne.0) go to 900
c
c     wait for results from nodes
c
      msgtyp  = 2
      len = 10
      do 30 i=1,numpro
        call pchrecv (i, msgtyp, buf, len, lenr, ierr)
```

```
c      ============
       if (ierr.ne.0) print *, 'error while recv. from node ', i
       if (ierr.ne.0) go to 900
       call pcunkr8 ( result(i), 1, 1, buf, len, ierr)
c      ============
       if (ierr.ne.0) print *, 'error while unpack. from node ', i
       if (ierr.ne.0) go to 900
       print *, 'I got',result(i), '  from node', i
       write(6,*) 'I got',result(i), '  from node', i
 30    continue
c ------ End user program ------------------------------
      stop
c
900   call pcperror('Host',ierr,msgtyp)
      stop
      end
```

```
      program slave1
c ********************************************************
c Example of fortran program illustrating the use of PCCOM
c "Node" or "Slave" program
c ********************************************************
      integer idfrom,msgtyp,len,ierr
      integer lenr, numpro, idpro, idhost
      real*8 buf(10), result, data(100)
      real*8 work
      include 'com486.h'
c     ==================
c
c Enroll this program in PCCOM
      ierr = 0
      open(99,file='slv.out')
      iopt = 2
c
c this is the first mandatory routine of PCCOM:
      call pcnopen (ierr)
c     ============
      if (ierr.ne.0) go to 900
c
c Get the own process ID
      call pcmypid( idpro )
c     ============
c Get the host ID
      call pchostid( idhost )
c     =============
c Get the number of processes
      call pcnumpro ( numpro )
c     =============
c ---- Begin user program ----------------------------
c     Receive data from host
      msgtyp  = 1
      len = 88
      call pcnrecv (idhost, msgtyp, buf, len, lenr, ierr)
c     ============
      if (ierr.ne.0) go to 900
c
      call pcunki4 ( n, 1, 1, buf, len, ierr)
c     ============
      if (ierr.ne.0) go to 900
      call pcunkr8 ( data, n, 1, buf, len, ierr)
c     ============
      if (ierr.ne.0) go to 900
c
      call pcbarr ( numpro, ierr)
c     ===========
      if (ierr.ne.0) go to 900
c
c     Do calculations with data
      result = work( idpro, n, data, numpro )
c     Send result to host
```

```
      msgtyp  = 2
      call pcinits ( buf, len)
c     ============
      call pckr8 ( result, 1, 1, buf, len, ierr)
c     ==========
      if (ierr.ne.0) go to 900
c
      call pcnsenh ( msgtyp, buf, len, ierr)
c     ============
      if (ierr.ne.0) go to 900
c ------ End user program ------------------------------
      stop
c
900   call pcperror('Slave',ierr,msgtyp)
      stop
      end
c
c **********************************************************************
      double precision function work( idpro, n, data, numpro )
c     include 'com486.h'
c     ================
c ****************************************
c     Just a simple routine for illustration
c ****************************************
      real*8  data(n), sum, psum, abf
      integer i, n, idpro, inum, numpro
      integer ierr, len, msgtyp
      sum = 0.0
      do 10 i=1,n
        sum = sum + idpro * data(i)
 10   continue
c     ------------------------------------------------
c     Pass partial result to neighboring node
c     to illustrate node-to-node communication
c     ------------------------------------------------
      msgtyp = 77
      inum   = idpro + 1
      if( idpro .eq. numpro ) inum = 1
c
c       send SUM
c
      call pcnsenn ( inum, msgtyp, sum, 1, ierr )
c
c       receive PSUM
c
      call pcnrecv ( -1, msgtyp, psum, 1, lenr, ierr )
      if (ierr.ne.0) go to 900
      work  =  sum  +  psum
      return
900   call pcperror('Work',ierr,msgtyp)
      stop
      end
```