Apéndice C: Comunicaciones en la Red Local del CeTAD

Las redes de interconexión son una parte esencial en la arquitectura de las computadoras paralelas. De hecho, una buena proporción de la bibliografía dedicada al cómputo paralelo está dedicada a este tema. Tiene relevancia desde dos puntos de vista fuertemente relacionados con las aplicaciones paralelas: flexibilidad y rendimiento. Además, se debe tener en cuenta que a mayor flexibilidad y rendimiento de las redes de interconexión se tendrá mayor costo de hardware, y que el crecimiento de este costo suele ser bastante más que lineal con respecto al crecimiento de la cantidad de procesadores de la máquina paralela.

En el caso particular de las multiplicaciones de matrices en paralelo que se ha analizado por experimentación, la red de interconexión es de fundamental importancia. Los índices de rendimiento muestran que la mayoría del tiempo de ejecución de la multiplicación de matrices en paralelo es utilizado para la transmisión de datos. Dado el impacto que el bajo rendimiento de las comunicaciones tiene sobre el rendimiento total de la aplicación paralela, es necesario tener una forma precisa de caracterizar el rendimiento de la red de interconexión para tomar las decisiones necesarias para la optimización del rendimiento de la aplicación completa.

Este Apéndice se dedica principalmente a la caracterización del rendimiento de la red de interconexión de la red local del CeTAD. Se desarrollan y se muestran los resultados de un conjunto de experimentos muy sencillos que permiten evaluar la red con bastante precisión. Si bien la biblioteca de pasaje de mensajes que se utiliza es PVM (Parallel Virtual Machine), se comentan algunas características de rendimiento que son comunes a la mayoría (sino *a todas*) las bibliotecas de pasaje de mensajes implementadas para redes locales de computadoras, tales como las implementaciones de MPI (Message Passing Interface). Es importante notar desde el principio mismo que se intenta conocer el rendimiento a nivel de las aplicaciones de usuario, ya que puede estar muy lejano de los valores definidos por (o *posibles* en) el hardware de comunicaciones.

Si bien se dan un conjunto bastante extenso de resultados de rendimiento de comunicaciones punto a punto (entre dos procesos ejecutándose en diferentes máquinas) en PVM, también se dan los resultados de rendimiento de los mensajes broadcast. De hecho, tal como se ha diseñado el algoritmo paralelo de multiplicaciones de matrices es el rendimiento de los mensajes broadcast el que realmente afecta el rendimiento final del procesamiento paralelo de multiplicación de matrices. Finalmente, se dan las características de rendimiento en las redes locales del LQT y del LIDI, donde se llevaron a cabo experimentos equivalentes, se comentan las diferencias más notables de rendimiento en estas redes y las razones por las cuales se dan estas diferencias.

C.1 Introducción

La red de interconexión de procesadores es elemental en las computadoras paralelas. En el caso de las computadoras paralelas con (o basadas en) memoria física compartida, esta red de interconexión se puede identificar claramente no en cuanto a la interconexión de los procesadores entre sí sino en cuanto a la interconexión de los procesadores con la memoria. Expresado de otra manera, en estas computadoras paralelas quitar la red de interconexión de los procesadores con la (*única*) memoria elimina completamente la posibilidad de ejecutar aplicaciones. Por otro lado, si a una computadora paralela con memoria distribuida se le quita la red de interconexión de procesadores, deja de ser una computadora paralela y se transforma en un conjunto de computadoras separadas o módulos de CPU-Memoria, sin la capacidad de cooperación para la resolución de un problema. Dado que las redes de computadoras utilizadas para cómputo paralelo son claramente pertenecientes a la clase MIMD de memoria distribuida, se continuará considerando a la red de interconexión para la transmisión de datos entre procesadores (o computadoras directamente).

En relación con la flexibilidad de una red de interconexión de procesadores, se busca no solamente que haya una forma de transferir datos entre dos procesadores sino que también se tenga el máximo de comunicaciones al mismo tiempo o simultáneas. Los ejemplos clásicos en este sentido se enfocan en la capacidad o no de que todos los pares posibles de procesadores se puedan comunicar al mismo tiempo, o que sea posible que un solo paso (o en una cantidad de pasos independiente de la cantidad de procesadores que se tengan interconectados) se pueda transferir información desde un procesador hacia todos los demás.

La flexibilidad que tenga una red de interconexión definirá la facilidad (o dificultad) de las aplicaciones de usuario para resolver la comunicación entre sus procesos. La idea subyacente es que nunca se debería perder de vista que cada uno de los procesadores será el encargado de la ejecución de uno o más procesos que se comunicará con otros procesos asignados a otro/s procesador/es.

La visión del rendimiento de una red de interconexión está directamente relacionada con el tiempo de transferencia de los datos entre los procesadores de una computadora paralela. Esta visión del rendimiento no necesariamente es disjunta de la flexibilidad. De hecho, a mayor cantidad de transferencias de datos simultáneas entre pares de procesadores será también mayor la capacidad o la cantidad de datos que una red de interconexión puede transferir por unidad de tiempo.

Si bien es importante esta idea de *ancho de banda* (tasa de transferencia) o cantidad de datos por unidad de tiempo, otro de los índices de rendimiento importantes es el tiempo mínimo de comunicación entre dos procesadores, o tiempo de inicialización de las comunicaciones (*startup*), o también llamado *latencia* de comunicación entre procesadores.

En términos de costo se tiene una relación invariante a través de las distintas posibilidades de redes de interconexión: a mayor flexibilidad y/o rendimiento de la red de interconexión el costo también aumenta. El crecimiento del costo varía según la red de interconexión que se utiliza, pero en muchos de los casos aumentar la cantidad de procesadores de la

computadora paralela implica un crecimiento más que lineal del costo de la red de procesadores. En el caso particular de las redes de computadoras instaladas, el costo es cero (en general, despreciable), dado que ya están interconectadas.

El principal inconveniente de las redes de interconexión de computadoras en cuanto a rendimiento es que no fueron diseñadas para cómputo paralelo. En este sentido, el rendimiento de las redes locales se ubica varios órdenes de magnitud por debajo de las redes de interconexión de las computadoras paralelas *tradicionales*. Es por eso que se torna muy importante evaluar su rendimiento desde el punto de vista de los procesos de usuario que componen una aplicación paralela.

Por otro lado, el rendimiento de la red de interconexión tiene relación directa con el rendimiento y con la granularidad de las aplicaciones paralelas que se pueden ejecutar sobre la computadora. Todo tiempo de comunicación tiende a degradar el tiempo total de ejecución de una aplicación paralela, a menos que se disponga y se aproveche al máximo la capacidad de solapar en el tiempo cómputo con comunicación. Desde el punto de vista de la granularidad, si el tiempo de comunicación para la obtención de un resultado en el procesador P_1 es igual o mayor que el tiempo de cómputo necesario para calcularlo, entonces lo más razonable es obtenerlo localmente (en P_1), ahorrando tiempo y/o complejidad de la aplicación.

C.2 Redes Ethernet

En el contexto particular de las redes de computadoras instaladas, redes locales o LAN (Local Area Networks), la red de interconexión más utilizada es la definida por el protocolo estándar IEEE 802.3. Este estándar es también conocido como red Ethernet de 10 Mb/s por su capacidad de transmisión de 10^6 bits por segundo. Las características de esta red de interconexión es muy bien definida y conocida en términos de hardware y de las características que hacen a su flexibilidad y rendimiento.

Además, la mayoría de las características de la red Ethernet de 10 Mb/s son similares a la red Ethernet de 100 Mb/s, donde se cambian solamente los parámetros/índices referidos a rendimiento o capacidad de comunicación. Esta similitud está ejemplificada en, y también aprovechada por muchas de las empresas de hardware de comunicaciones que se encargan de comercializar placas de interfase de comunicaciones (NIC: Network Interfase Card) con ambas capacidades de comunicación y denominadas de 10/100 Mb/s.

La Figura C.1 muestra esquemáticamente la forma lógica básica en que se conectan las estaciones de trabajo en una red local utilizando Ethernet. Se puede notar fácilmente que es de tipo bus, donde las características principales de cada transferencia de datos son:

- no se manejan prioridades ni es predecible el tiempo de acceso al medio,
- tiene un único emisor,
- ocupa el único canal de comunicaciones,
- puede tener múltiples receptores,
- el modo de acceso al medio es CSMA/CD (Carrier Sense, Multiple Access / Collision

Detect).

Las dos primeras características implican claramente que no puede haber más de una transferencia de datos simultáneamente, porque de hecho hay un único canal de comunicaciones que es compartido por todas las computadoras. La última característica enunciada hace muy natural la implementación de las comunicaciones del tipo *broadcast* y/o *multicast*, donde desde una computadora se emite un mensaje que es recibido en todas las demás o en un subconjunto de las demás de la red respectivamente.

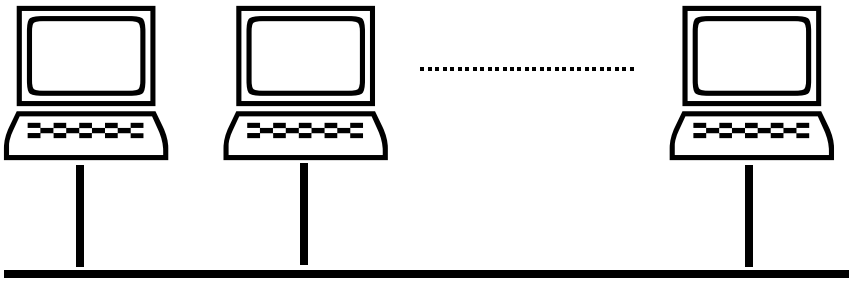


Figura C.1: Red Ethernet.

El hardware de comunicaciones inicialmente adoptado en la mayoría de las instalaciones estuvo basado en cables coaxiales, con lo cual se logra que la topología física sea igual a la topología lógica de la Figura C.1.

Gradualmente, el cableado (*wiring rules*) utilizado en la mayoría de las instalaciones se ha cambiado hacia la utilización del cable de par trenzado con *hubs* (básicamente concentradores y repetidores de comunicaciones) y con *switches* de comunicaciones, que tienen la capacidad de aislar las comunicaciones que son punto a punto. Esta aislación en los *switches* se produce cuando el hardware detecta que hay transmisión de datos punto a punto entre dos de las computadoras que están interconectadas por un *switch*. También son posibles varias combinaciones de *hubs* y *switches* con el objetivo de mantener un cierto rendimiento a medida que el tráfico de datos aumenta y también reducir el costo al evitar la utilización masiva de *switches* de comunicaciones.

Estas redes de comunicación son importantes no solamente por la cantidad de instalaciones actualmente funcionando sino porque son claramente menos costosas que las demás alternativas que se comercializan. Son menos costosas en el hardware necesario (placas, conectores y cables) y en lo referente a instalación: desde mano de obra (técnicos) hasta reconocimiento y puesta en marcha del hardware por parte del sistema operativo. Todo esto necesariamente reduce los costos de instalación y de mantenimiento de las redes Ethernet.

La reducción de costo con respecto a las demás alternativas de interconexión de computadoras implica una gran inercia en el mantenimiento de las redes Ethernet así como también en la instalación de nuevas redes con este hardware. Siempre se debe tener en cuenta que el costo puede incluir aspectos como: placas de red en cada computadora, cableado (que puede incluir hubs y/o switches en el caso de las redes Ethernet), instalación, mantenimiento y personal técnico capacitado.

C.3 Evaluación del Rendimiento

El tiempo de comunicación (utilizado en general para caracterizar el rendimiento de una red de interconexión) entre dos procesadores en general se caracteriza con [7] [11]

$$t(n) = \alpha + \beta n \qquad \text{C.1}$$

donde

- n es la unidad de información que se transfiere y que se desea medir (bit, byte, representación de un número en punto flotante con precisión simple, etc.),
- α es el tiempo necesario para establecer la comunicación entre los dos procesadores, que se suele llamar *tiempo de latencia* de la comunicación. Es básicamente el tiempo mínimo que toda comunicación entre dos procesadores utilizará independientemente de la cantidad de información que se transfiera. Normalmente está dado por el hardware de comunicaciones y puede estimarse con el tiempo que toma transferir una unidad de información o, en el caso en que sea posible, un mensaje sin datos.
- β es el valor inverso del ancho de banda asintótico o tasa de transferencia de datos de la red de comunicaciones, es decir que $1/\beta$ es el ancho de banda asintótico. Normalmente, la tasa de transferencia de datos de la red de comunicaciones está dada por la cantidad de datos (bits, bytes, etc.), por unidad de tiempo que pueden transmitirse entre dos procesadores. Normalmente tiene un límite mínimo dado por el hardware de comunicaciones a lo que se debe agregar el tiempo que consumen los procesos y/o funciones del software de comunicación.

Si bien el hardware de comunicaciones o red de interconexión de procesadores tiene valores bien definidos para los parámetros α y β, en general el usuario suele obtener valores peores desde el punto de vista del rendimiento de la red de interconexión de procesadores que los dados por el hardware. Tanto el tiempo de latencia como el tiempo que lleva transmitir cada unidad de información se ven afectados (y por lo tanto aumentan) cuando en la transferencia intervienen todas las capas de comunicación necesarias para que un mensaje de un proceso de usuario ejecutándose en un procesador llegue a otro proceso de usuario ejecutándose en otro procesador.

Es también muy difícil realizar una estimación *a priori* de la sobrecarga que las bibliotecas de comunicaciones que tienen disponibles los (procesos de) usuarios, la interfase del sistema operativo con el usuario y/o con las bibliotecas mencionadas anteriormente y los protocolos de comunicaciones, por ejemplo, imponen a las comunicaciones que se llevan a cabo físicamente sobre el medio de comunicación que se utilice. Por esta razón son bastante utilizados los métodos experimentales de medición de los parámetros α y β reales que las aplicaciones de usuario encontrarán respecto a la red de interconexión de la computadora paralela.

En el contexto específico de las redes de computadoras, y con la posibilidad de hardware heterogéneo, esta sobrecarga se torna más importante cuando se debe evaluar el rendimiento de la red de interconexión de procesadores. En el caso de las computadoras paralelas tradicionales, el cálculo de α y β reales suele ser mucho más sencillo y con

valores finales (los que los procesos de las aplicaciones paralelas de usuario obtienen) más cercanos a los del hardware porque desde el hardware de comunicaciones hasta la interfase que utilizan las aplicaciones de usuario está *todo* orientado a hacer cómputo paralelo.

Es muy difícil cuantificar de manera exacta la relación de las redes locales con respecto a las redes de interconexión de las computadoras paralelas tradicionales en términos de los índices de rendimiento mencionados. Lo que es generalmente aceptado es que la peor relación está dada con respecto al tiempo de inicialización de los mensajes a comunicar entre dos procesadores. Más aún, en el contexto de las redes locales heterogéneas el tiempo de inicialización de las comunicaciones suele depender de las computadoras utilizadas dado que están involucrados los tiempos de llamadas al sistema operativo y su consiguiente sobrecarga en cuanto al mantenimiento y manejo de los protocolos (o *pila de protocolos*) utilizados. En un nivel más cercano al hardware, también están involucrados los tiempos de

- acceso a memoria,
- inicialización-utilización de canales de DMA (Direct Memory Access) en caso de ser utilizados y
- manejo de interrupciones relacionadas y/o interfase con la placa de red de cada computadora a comunicar.

Resumiendo, el rendimiento de las redes locales es inferior al de las redes de interconexión de procesadores de una máquina paralela tradicional desde varios puntos de vista:

- Latencia y ancho de banda.
- Capacidad de solapamiento.
- *Heterogeneidad* de latencia dependiendo de la heterogeneidad de las máquinas.

C.4 Evaluación con el Método de ping-pong

En el contexto de las computadoras paralelas con arquitectura de base MIMD, el método experimental para evaluar el rendimiento de la red de interconexión de procesadores o, más específicamente, los valores reales de los parámetros α y β, ha sido el de los mensajes *pingpong*. En sí mismo, el método es muy sencillo, dado que para evaluar el tiempo de comunicación entre dos procesadores P_1 y P_2, los pasos a seguir son los que muestra la Figura C.2:

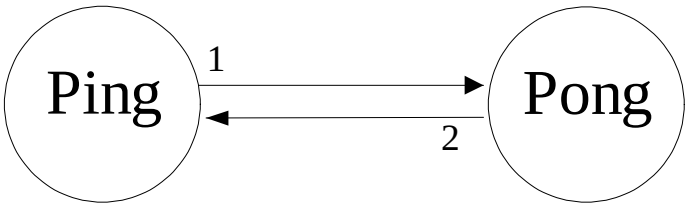


Figura C.2: Procesos *Pingpong*.

1. Enviar un mensaje desde el procesador P_1 al procesador P_2
2. Enviar el mensaje recibido en el procesador P_2 al procesador P_1 nuevamente

3. En el procesador P_1 se conoce el tiempo total utilizado para la comunicación de los dos mensajes y, por lo tanto, se divide ese tiempo por dos llegándose así al tiempo de comunicación del mensaje en una de las direcciones.

Una de las características más atrayentes de este método, además de su sencillez, consiste en que no necesita la sincronización de ninguno de los procesadores que intervienen en la transferencia de información para obtener un tiempo confiable de comunicación de datos. De otra manera, se debería conocer el tiempo de envío del mensaje desde el procesador P_1 al procesador P_2, el tiempo de llegada del mensaje al procesador P_2 y los procesadores deberían estar sincronizados con respecto a su registro de tiempo.

Otra de las ventajas del método consiste en la independencia de la forma de comunicar mensajes entre los procesadores. El tiempo que lleva enviar y recibir el mismo mensaje mensaje en P_1 (comunicándose con P_2) se puede tomar independientemente de la forma en que el mensaje sea transmitido. De hecho, se pueden analizar distintas alternativas disponibles de comunicación entre los procesadores para elegir la más conveniente.

Por otro lado, una de las restricciones respecto a la confiabilidad de este método recae en que el tiempo de comunicación entre los procesadores no debe variar según la *dirección* de la comunicación, es decir que hay cierta *simetría* de comunicaciones entre los procesadores. En la enumeración anterior, se asume que el tiempo para enviar un mensaje desde P_1 a P_2 es el mismo que el de enviar el mismo mensaje de P_2 a P_1. De todas maneras, esta situación es sin lugar a dudas mayoritaria en el campo de las redes de interconexión en general.

Otra de las simplificaciones del método *pingpong* consiste en enmascarar, o mejor expresado, no tener en cuenta la posibilidad de hacer transmisión y recepción de datos simultáneamente (comunicación *full duplex*), o la posibilidad de solapar cómputo con comunicación que las aplicaciones podrían aprovechar. En ambos casos, es decir con el hardware disponible para hacer una o ambas cosas, si se obtienen los tiempos de comunicación punto a punto es posible llegar a los valores de rendimiento de comunicación que las aplicaciones pueden obtener.

En el caso específico de las redes de computadoras, el usuario normalmente no tiene demasiado control (o ningún control) sobre el aprovechamiento o no de las capacidades del hardware. Por lo tanto, los valores de rendimiento que se obtienen por el método del pingpong serán los más cercanos a lo que las aplicaciones (paralelas) de usuario podrán obtener.

C.5 Distintas Formas de Transmisión de Mensajes con PVM

Dado que es necesario caracterizar el tiempo de comunicación entre los procesos de un programa paralelo y que éstos se comunican utilizando las rutinas de comunicación provistas por PVM, es necesario explorar todas las alternativas posibles en cuanto al

rendimiento alcanzable con estas rutinas para las aplicaciones de usuario.

Con el método de pingpong explicado previamente se puede evaluar bastante rápidamente cuál es el rendimiento obtenible en las comunicaciones entre los procesos que se asignan a los distintos procesadores de la máquina (*virtual*) paralela. Además, dado que se tienen los valores relacionados con el rendimiento de la red de comunicaciones (α y β), del hardware de comunicaciones se puede conocer con certeza el grado de sobrecarga de tiempo por utilizar PVM para comunicar tareas.

Cuando se debe evaluar el rendimiento de las comunicaciones teniendo en cuenta no solamente el hardware sino también el software de comunicaciones (básicamente los procesos del sistema operativo involucrados más las rutinas de comunicaciones de PVM) es necesario explorar las distintas alternativas de comunicación para transferir datos entre los procesos de la aplicación paralela.

En general, PVM tiene dos niveles de flexibilidad cuando se deben transferir datos entre los procesos: codificación de datos y "ruteo" (en terminología de PVM) de los mensajes. La codificación de datos tiene relación con la representación de información que se hace en cada procesador (computadora) y el ruteo de los mensajes se relaciona con la forma en que los datos se transfieren entre los procesos de la aplicación paralela utilizando la red de física de comunicaciones y las rutinas/procesos de comunicación de PVM. En las subsecciones que siguen se detallan las alternativas para codificación y ruteo en PVM. En el caso particular de la codificación, se describirá también una alternativa a las formas clásicas que se utilizan tanto en PVM como en MPI, que se denominará *Traducción Directa de Representación de Datos*.

Es necesario aclarar que si bien esta descripción es específica de PVM, tanto en MPI como en cualquier otra biblioteca que se utilice para hacer cómputo paralelo en redes de computadoras será necesario definir tanto la forma en que las distintas representaciones de datos se compatibilizan como la forma en que los datos se transfieren sobre la red de interconexión de computadoras.

C.5.1 Codificación de los Datos de un Mensaje en PVM

La codificación de datos básicamente se debe elegir entre lo que en PVM se denomina:

a) *PvmDataDefault*: se utiliza cuando los procesos que se comunican están asignados a procesadores con distintas arquitecturas o cuando la aplicación no tiene ningún conocimiento de la arquitectura de los procesadores (computadoras) sobre la que se ejecuta. Los datos a transferir entre los procesos son codificados en formato XDR antes de ser enviados, luego se copian a un área de memoria desde donde las rutinas de PVM harán el envío (buffers) de información y luego serán decodificados (del formato XDR) al ser recibidos antes de ser utilizados por el proceso receptor.
b) *PvmDataRaw*: se utiliza cuando la arquitectura de la máquina paralela es homogénea, sea una red de computadoras o sea un multiprocesador. Los datos que se transfieren entre los procesos no se codifican de ninguna manera, solamente se copian a los buffers de PVM antes de hacer el envío por la red de comunicaciones.
c) *PvmDataInPlace*: Esta alternativa es similar a *PvmDataRaw* pero sin copia de los datos

de usuario a los buffers. No hay codificación de los datos, ni gasto extra de memoria por la comunicación de datos, ni tiempo de copia de información, pero el usuario debe asegurar que los datos no son cambiados desde que se efectúa el llamado a la rutina de envío hasta que los datos son enviados efectivamente al proceso destino.

Con la Figura C.3 se pueden mostrar esquemáticamente los tres tipos de codificación y su relación con la codificación y la memoria utilizada para el envío de datos desde el proceso P_1 asignado a un procesador y el proceso P_2 asignado a otro.

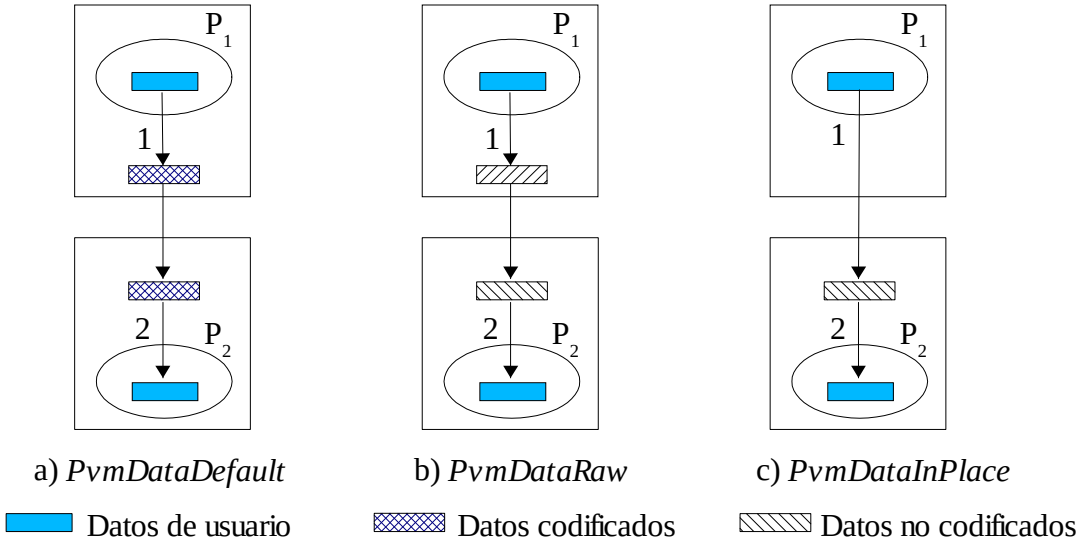


Figura C.3: Alternativas de Codificación en PVM.

Cuando se utiliza codificación a) *PvmDataDefault*, los datos son copiados desde el área de datos de proceso usuario P_1 al área de buffers de PVM y codificados en formato XDR [10] en el paso 1. Luego estos datos codificados son enviados al procesador destino (en el cual está asignado el proceso receptor P_2), utilizando la red de interconexión, donde se reciben sobre otra área de buffers de PVM. En el paso 2 se decodifican y se copian los datos ubicados en los buffers de PVM al área de datos del proceso receptor P_2.

Cuando se utiliza la codificación b) *PvmDataRaw*, el proceso es igual, con la excepción de no utilizar ningún tipo de codificación para los datos que se envían. La secuencia de bytes que estaban en el área de datos del proceso P_1 llegan al proceso P_2. Con este método se se ahorra en memoria de los buffers utilizada por la codificación XDR y también se evita el uso de CPU que tal codificación involucra, pero las copias y los buffers son utilizados de la misma manera que en el caso de utilizar la codificación PvmDataDefault.

Cuando se utiliza la codificación c) *PvmDataInPlace*, se evita no solamente lo relativo a la codificación XDR (procesamiento y memoria asociada) sino también toda la memoria necesaria para almacenar el mensaje en el procesador origen de la comunicación. En el caso de la Figura C.3, en el primer paso se envían los datos desde el proceso de usuario P_1 que envía el mensaje al procesador al cual está asignado el proceso receptor P_2, donde se almacena en los buffers de PVM. Expresado de otra manera, ya no son necesarios los buffers de PVM de salida para el mensaje.

En los ejemplos de evaluación del rendimiento de las comunicaciones que se incluyen en la distribución de PVM, la forma de codificación de datos utilizada es *PvmDataRaw*, con un comentario indicando que en caso de haber heterogeneidad en la máquina paralela se debe cambiar esta forma de codificación a *PvmDataDefault*. En las redes de computadoras en general, no queda más alternativa que la de codificar los datos (con el agregado de las copias a buffers de PVM) con esta forma *PvmDataDefault* para que no pierdan su significado al ser transferidos a otro proceso que se ejecuta sobre otro procesador (computadora). Una vez que se acepta la heterogeneidad de las computadoras utilizadas para cómputo paralelo no es posible asumir que la representación de los datos en todas las computadoras es homogénea.

Desde el punto de vista del rendimiento, el método de codificación más apropiado es el *PvmDataInPlace*. La pérdida de flexibilidad por la restricción en cuanto a la imposibilidad de modificación de los datos que se envían no parece ser un problema en el caso particular de la multiplicación de matrices. Los datos que se envían (submatrices de la matriz B, en la operación $A = B \times C$) son de sólo lectura para todos los procesos que los utilizan. De todas maneras, el problema de la heterogeneidad de la representación de datos persiste.

Todas las aplicaciones numéricas dependen, en cuanto a representación de datos, de una cantidad bastante reducida de tipos de datos que son generalmente predefinidos por los lenguajes que se utilizan (normalmente C o FORTRAN). En general, se pueden identificar dos tipos básicos a partir de los cuales se definen las estructuras de datos con las cuales se opera en las aplicaciones (mayoritariamente vectores y matrices): punto flotante de precisión simple y punto flotante de precisión doble. La representación de números complejos, por ejemplo, necesarios también para una amplia gama de aplicaciones numéricas, se define en función de un par de números de punto flotante en precisión simple o doble, según la aplicación específica.

Como se puede notar a partir de las tres formas de codificación explicadas, la estrategia general para transferir datos de una computadora a otra es, esquemáticamente:

Figura C.4: Estrategia de De/Codificación de PVM/MPI.

Es decir que, antes de transferir datos a de una estación a otra los datos se codifican para conservar la información que representan. En este sentido, tanto PVM como MPI son similares, si hay codificación de datos para su transferencia esta codificación se lleva a cabo antes de hacer la transferencia.

De alguna manera, esta estrategia de codificación-transmisión-decodificación puede ser considerada conservadora o anticipada a la comunicación, porque siempre funciona de

forma independiente de las computadoras que se comunican y se lleva a cabo antes de que los datos se transfieran.

C.5.2 Traducción Directa de Representación de Datos

Una alternativa a esta forma anticipada de PVM (y MPI) de conservar la consistencia de los datos entre distintas computadoras consiste en retardar la de/codificación de forma tal que:

- Los datos son siempre enviados sin ningún tipo de codificación previa (ni a XDR ni a ningún otro formato).
- Los datos son recibidos como una secuencia de bytes en el receptor, junto con un descriptor del tipo de datos que representan más el tipo de arquitectura origen (tipo de representación de origen).
- Si el procesador donde está ubicado el proceso receptor tiene distinta representación de datos que el procesador donde está ubicado el proceso que envía, los datos son "decodificados": se cambia de la representación de datos origen a destino.

Por lo tanto, en realidad no hay de/codificación, sino que en el procesador destino, las rutinas de comunicación se encargarán de "traducir" la representación de datos de una computadora (desde donde se envió el mensaje) a la otra (donde se recibe el mensaje).

En el caso de codificar a una representación conocida se tienen dos traducciones: de la representación origen (desde donde el mensaje es enviado) a la representación conocida y luego de ésta a la representación destino (en donde el mensaje es recibido). En el caso de la traducción directa se tiene a lo sumo una sola, que cambia de la representación de datos origen a la representación de datos destino cuando es necesario.

Se debe notar que la traducción directa se debe hacer en el destino y no en el origen de cada mensaje, dado que de esta forma se evitan problemas con las llamadas comunicaciones colectivas como *multicast* y *broadcast*. En este tipo de comunicaciones desde un mismo proceso en un procesador se pueden enviar datos a múltiples procesos (lo que implica potencialmente procesadores destino), y por lo tanto no habría una única traducción posible (habría tantas como posibles destinos).

La traducción directa se puede hacer transparente para los procesos de usuario tanto como lo es la codificación XDR usada en PVM y en (algunas implementaciones de) MPI. En el caso de PVM no hay ningún inconveniente para su implementación porque se tienen funciones de identificación de cada computadora de la máquina paralela así como también formas de identificar la ubicación de cada tarea (procesador en el que se ejecuta).

Respecto de la codificación de los datos para su transmisión, la traducción tiene la ventaja de minimizar el procesamiento y la memoria utilizada para cada mensaje en el lado (procesador) del proceso que envía. De hecho, los datos se pueden tomar de la memoria del proceso origen, agregar los descriptores necesarios (que de todas maneras están en la codificación) y enviar el mensaje. En el lado (procesador) del proceso que recibe, tanto la memoria como el procesamiento dependen de la complejidad de la traducción de los datos. Como se verá a a continuación, como mínimo en el contexto particular de las redes de computadoras, la traducción de representación es muy sencilla, y por lo tanto también se

reduce la necesidad de memoria y procesamiento.

Avanzando en el análisis de la representación de números en el contexto particular de las computadoras, se encuentra una sorprendente homogeneidad en cuanto a la adhesión al estándar IEEE 754 [6] para la representación de números en punto flotante. En todas las computadoras a las cuales se tiene acceso y sobre las que se llevó a cabo la experimentación, que incluye

- PCs con procesadores Pentium (en alguna de sus múltiples versiones), Celeron, y AMD K6-II,
- Estaciones de trabajo Sun con procesadores MicroSPARC-II,
- Una estación de trabajo IBM RS/6000, con procesador PowerPC,

la representación de punto flotante adoptada (a nivel de operaciones de la unidad de punto flotante del procesador) es la misma: IEEE 754 en ambas versiones (simple y doble).

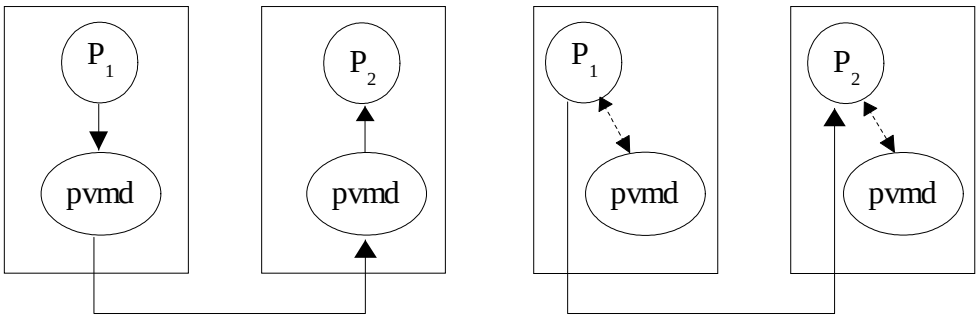
Se debe destacar que la heterogeneidad a nivel de representación de datos puede ser mayor en otros ámbitos, por ejemplo entre distintas máquinas paralelas tradicionales con unidades de punto flotante ampliamente más complejas. Independientemente de esto, se debe recordar que la tendencia (aún en las máquinas paralelas más potentes/costosas) es la utilización de hardware estándar. Las IBM SP2, por ejemplo, están basadas en PowerPCs, y las ASCI Red y Blue están basadas en Pentium Pro y MIPS R1x000 respectivamente.

Aún cuando todas las computadoras adoptan IEEE 754 como su representación de números en punto flotante, esto no significa que al enviar una secuencia de bytes desde una computadora a otra estos bytes significarán (representarán) lo mismo en ambas. Un escollo más a superar es la forma de almacenamiento de los bytes de un tipo particular de datos (en el caso que se viene analizando: números en punto flotante) en la memoria. En este caso las "normas" que se siguen son dos y de hecho no hay muchas más alternativas para explorar: primero el byte más significativo (llamada usualmente *little endian* en la literatura) o primero el byte menos significativo (*big endian* en la literatura). Es claro que la traducción de un formato a otro es inmediata y sin grandes requerimientos de memoria ni de procesamiento.

De los dos párrafos anteriores se deduce que, como mínimo para la representación de números en punto flotante, la traducción directa de las representaciones de datos entre computadoras es ventajosa con respecto a la codificación, tanto en requerimientos de memoria como de procesamiento. No es difícil hacer un análisis similar con los demás tipos básicos de datos (caracteres, número enteros, etc.) y llegar a la misma conclusión. Es por esta razón que la experimentación incluye esta forma de "codificación" de datos para la transmisión de mensajes entre los procesos *pingpong*.

C.5.3 Ruteo de los Datos de un Mensaje en PVM

El ruteo de mensajes en PVM se refiere a la forma en que los datos de un mensaje se transportan entre los procesos de usuario y el proceso mismo de PVM (*pvmd*) en cada computadora. Las dos formas más comunes de ruteo de los mensajes entre las tareas son esquematizadas en la Figura C.5.



a) PvmRouteDefault b) PvmRouteDirect

Figura C.5: Alternativas Básicas de Ruteo en PVM.

En el caso a) *PvmRouteDefault*, los datos se transfieren entre los procesos *pvmd* sobre la red de interconexión de computadoras utilizando el protocolo de comunicaciones UDP (sobre IP).

En el caso b) *PvmRouteDirect*, los datos se transfieren directamente entre los procesos de la aplicación paralela sobre la red de interconexión de computadoras utilizando el protocolo de comunicaciones TCP (también sobre IP). La recomendación general que se hace en la documentación de PVM es que, en términos de rendimiento, la opción b) es la mejor.

C.6 Distintas Formas de Transmisión de Mensajes con MPI

Dado que MPI se propone como una biblioteca estándar de pasaje de mensajes entre procesos de una aplicación paralela, detalles de opciones tales como la codificación y el ruteo (en el sentido de PVM) no están disponibles a nivel del usuario. En este sentido, la creación de MPI es bastante lejana e independiente de las redes de computadoras y por lo tanto la heterogeneidad en la representación de datos o las formas alternativas a nivel de protocolos de comunicación entre procesadores no tienen la relevancia que adquieren desde el principio en PVM que fue creado para redes de computadoras (heterogéneas). En términos de las posibles implementaciones, está ampliamente demostrado que MPI es posible en todo el rango de computadoras paralelas de pasaje de mensajes, sean multiprocesadores, multicomputadoras con redes de interconexión *ad hoc* para cómputo paralelo o redes de computadoras.

Las implementaciones de MPI para redes de computadoras normalmente asumen que la representación de datos es heterogénea (de hecho no les queda mucha alternativa) y conectividad IP. El método de *codificación anticipada* (con XDR por ejemplo), suele utilizarse para resolver la heterogeneidad de la representación de datos. Como siempre en el contexto de MPI debe recordarse que es la implementación la que toma una decisión de este tipo y, por lo tanto, distintas implementaciones de MPI pueden tener distintas formas de implementarlo. De forma similar, lo que en PVM se denomina ruteo (referente a la

forma en que los datos de un mensaje son enviados desde el proceso emisor al proceso receptor), depende de la implementación de MPI, aunque la mayoría coinciden en utilizar conectividad IP (al menos las libres que pueden obtenerse vía Internet).

En todo caso, independientemente de la implementación de MPI, sea para redes de computadoras o para cualquier otro tipo de arquitectura de cómputo paralelo, no se tienen alternativas a nivel de usuario sobre la codificación ni sobre el ruteo de los datos de los mensajes. En MPI se gana en transparencia respecto de la implementación y de la arquitectura paralela y en el caso particular de las redes de computadoras quizás se pierde rendimiento si se tiene homogeneidad de máquinas o si se utiliza la traducción de las representaciones de los tipos de datos explicada previamente.

C.7 Experimentación Inicial con PVM

La experimentación inicial con PVM se llevó a cabo para tener una base de cálculo de α y β utilizando las distintas formas de codificación y ruteo que en PVM se pueden seleccionar a nivel de usuario. Las máquinas con las cuales se llevó a cabo la experimentación se detallan en el Apéndice X: computadoras del CeTAD, interconectadas con una red Ethernet de 10 Mb/s. Dado que las computadoras **cetadfomec1** y **cetadfomec2** son exactamente iguales se muestran los resultados para una de ellas (**cetadfomec1)**, que por razones de espacio en los gráficos se abrevia como **cf1**.

Se utilizó el método de ping-pong ubicando al proceso *ping* siempre en una misma computadora y midiendo los tiempos con el proceso *pong* en cada una de las demás por vez. Si bien lo más importante a estimar (y por lo tanto a medir) son los parámetros α y β, en el caso específico de las redes de computadoras puede ser útil el tiempo necesario para resolver la heterogeneidad de representación de datos (de/codificación o traducción). La computadora elegida para ubicar el proceso ping es **purmamarca** dado que tiene la cantidad de memoria necesaria para todas las longitudes de mensajes (64 MB) y que es la más rápida de todas las computadoras del CeTAD.

Las mediciones se llevaron a cabo con la red libre de interferencias (sin más tráfico en la red del que las computadoras generaban por el pingpong), y de hecho se hicieron en diferentes días bajo las mismas condiciones, obteniendo los mismos resultados.

De acuerdo a lo que se ha explicado, en las redes de computadoras heterogéneas se tienen cuatro alternativas para el manejo de los mensajes entre procesos

1. Ruteo *PvmRouteDefault* con Codificación *PvmDataDefault*, es decir transmitir datos codificados en formato XDR utilizando el proceso pvmd en cada una de las computadoras involucradas.
2. Ruteo *PvmRouteDefault* y sin codificación sino con traducción de la representación de los datos, es decir traduciendo la representación siempre que sea necesario en el proceso destino y utilizando el proceso pvmd en cada una de las computadoras involucradas.
3. Ruteo *PvmRouteDirect* con codificación *PvmDataDefault*, es decir transmitir datos codificados en formato XDR directamente entre las tareas *ping-pong*.

4. Ruteo *PvmRouteDirect* y sin codificación sino con traducción de la representación de los datos, es decir traduciendo la representación siempre que sea necesario en el proceso destino y con los datos transferidos directamente directamente entre las tareas *ping-pong*.

Dado que es difícil encontrar un conjunto de longitudes de mensajes que sea representativo para todas las posibilidades de aplicaciones y paralelización, las longitudes elegidas cubren un amplio rango: desde mensajes de ocho bytes (los necesarios para dos números en punto flotante de precisión simple o uno de precisión doble) hasta mensajes de 10^7 bytes. Las longitudes intermedias elegidas son 10^2, 10^3, $6x10^4$, 10^6 y 10^7. El caso particular de la longitud $6x10^4$ (que no sigue exactamente la "convención logarítmica" de crecimiento de longitud de mensajes) fue elegido para poder luego comparar los resultados con la versión de *pingpong* del sistema operativo (en particular, el de Linux): el comando *ping*, cuya justificación de utilización se describe en la siguiente sección. Dado que la longitud $6x10^4$ es muy cercana a 10^5 como para que ésta proporcione datos significativos, se decidió no incluirla en el reporte de resultados.

Para mayor información y claridad de los datos obtenidos, se presentarán los resultados en dos formatos:
1. Tiempo total de comunicación para el *pingpong*. Esta alternativa se presenta a su vez en formato logarítmico de los tiempos, dadas las longitudes de mensajes elegida.
2. Ancho de banda o bytes/segundo, para una mejor idea de la relación de rendimiento entre lo que se obtiene a nivel de usuario con PVM y el hardware (Ethernet de 10 Mb/s).

C.7.1 Rendimiento con Ruteo entre pvmds y con Codificación

Se supone que esta alternativa es la menos conveniente en cuanto a rendimiento de la red de interconexión de computadoras, y los resultados obtenidos en términos de tiempo de comunicaciones (pingpong) se muestran en la Figura C.6. Se puede ver claramente que el tiempo de latencia de las comunicaciones en PVM entre procesos varía entre 1 y 10 ms dependiendo de la computadora, dado que estos valores inicialmente se repiten a pesar de que las longitudes de mensajes varían entre 8 y 1000 bytes (dos órdenes de magnitud, en términos de crecimiento).

La escala logarítmica de tiempos de la Figura C.6 de alguna manera enmascara las diferencias entre computadoras, aunque se puede ver que para todas las longitudes se tienen diferentes tiempos de comunicación para las diferentes máquinas involucradas.

La Figura C.7 muestra los mismos resultados, pero en función de MB/s (megabytes por segundo), es decir de cantidad de información (2^{20} bytes) por unidad de tiempo. Tanto en esta figura como en todas las siguientes que se expresan en términos de MB/s se pueden visualizar mejor las diferencias relativas entre computadoras, así como también la relación con la capacidad del hardware. En el caso particular de esta figura, para todas las computadoras se puede comprobar lo que era de esperar: a mayor longitud de mensaje se obtiene un rendimiento mayor.

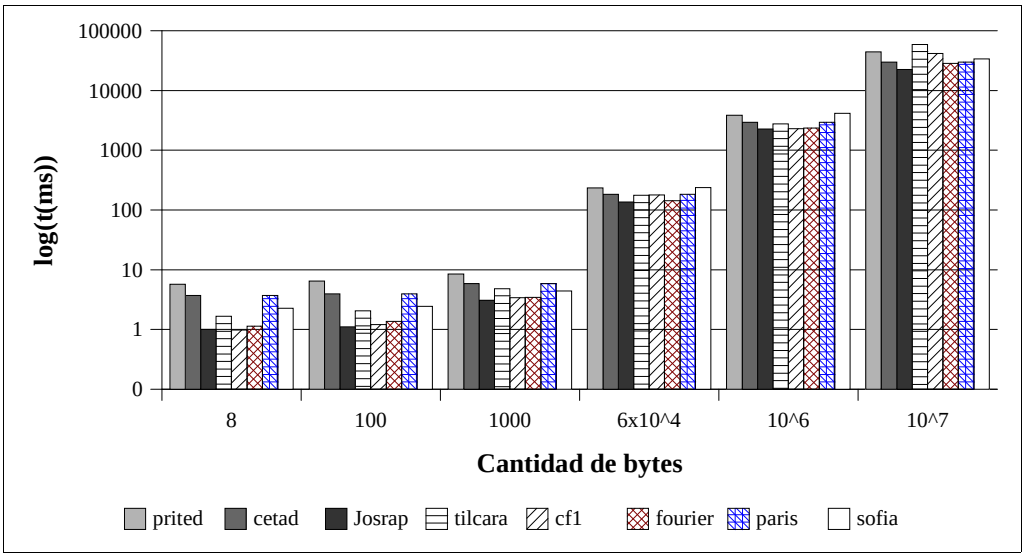


Figura C.6: Tiempos con Ruteo y Codificación *PvmDefault*.

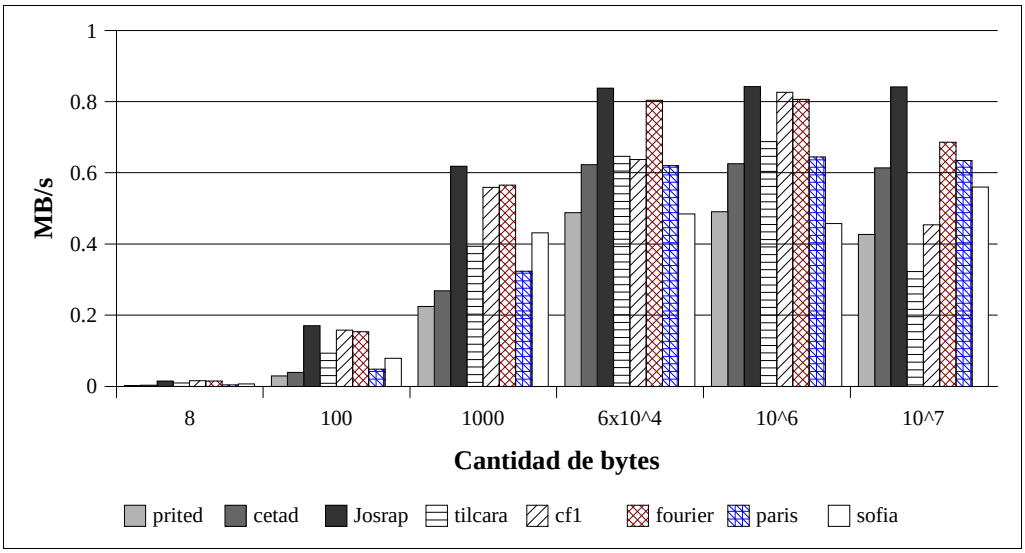


Figura C.7: MB/s con Ruteo y Codificación *PvmDefault*.

En el caso particular de **tilcara**, y **cf1** parece haber un comportamiento anómalo para los mensajes de 10^7 bytes, pero es explicable en función de sus tamaños de memoria principal de 32 MB. Este tamaño es insuficiente para el mensaje, más los buffers de PVM, más los demás procesos (pvmd, sistema operativo, etc.) y sus requerimientos de memoria. En ambos casos se debe recurrir al espacio de *swap* (manejado por el sistema operativo) y en ambos casos también esto produce una degradación notable a nivel de los procesos de usuario del rendimiento. En cierta forma, **prited** también tiene el mismo problema (pérdida de rendimiento por sus 32 MB de memoria principal), pero la pérdida relativa es mucho menor dado que en ningún caso llega a superar los 0.5 MB/s. Las demás computadoras tienen 64 MB de memoria principal o más y por lo tanto no llegan a tener la necesidad de recurrir al espacio de *swap*.

También a partir de la Figura C.7 se puede concluir que:

- existen claras diferencias de rendimiento entre las distintas computadoras, entre poco más de 0.4 MB y poco más de 0.8 MB/s,
- lo mejor que se obtiene es un poco más de 0.8 MB/s para tres de las computadoras (contando **cetadfomec2**, representada con **cf1**),
- el mejor rendimiento obtenido para cada una de las computadoras se logra con longitud de mensajes del orden de 10^4 bytes o más (representado con $6x10^4$ en la figura).

C.7.2 Rendimiento con Ruteo entre pvmds y con Traducción de Representación

La Figura C.8 muestra los resultados obtenidos en términos de tiempos absolutos de comunicación. Comparativamente con los que se muestran en la Figura C.6, los resultados de esta alternativa son similares en términos generales: latencia y tiempos absolutos, lo cual da una idea de la importancia relativamente baja de la forma de codificación de datos (o traducción de la representación, en este caso).

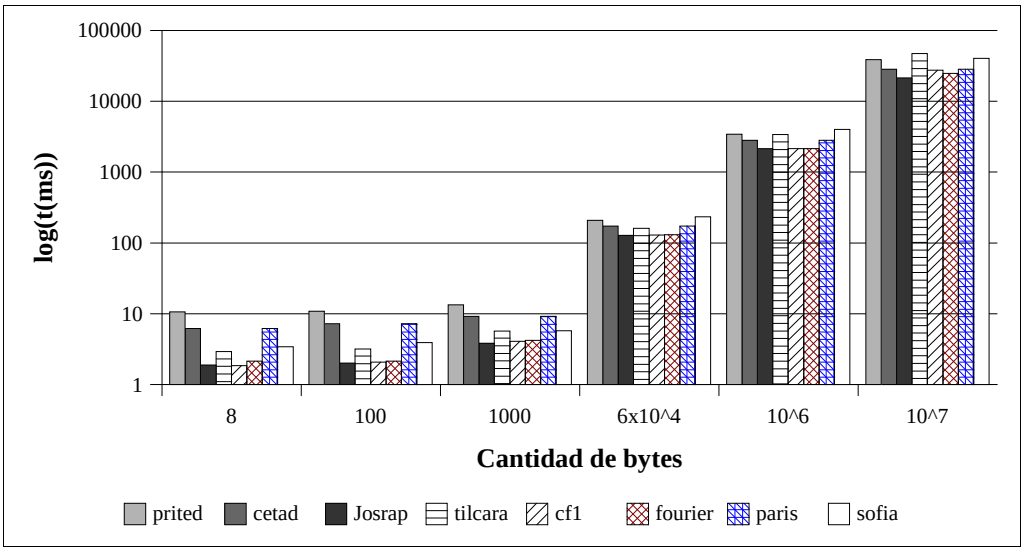


Figura C.8: Tiempos con Ruteo *PvmDefault* y Traducción de Representación.

La Figura C.9 muestra los mismos resultados pero en función de MB/s de acuerdo a los tiempos de comunicación y la cantidad de bytes que se transmiten según la longitud del mensaje. Se puede notar que la disminución en los requerimientos de memoria hacen que las máquinas con 32 MB (**tilcara**, por ejemplo) no reduzcan tan drásticamente su rendimiento cuando los mensajes son de 10^7 bytes, porque hacen un menor uso del espacio de *swap* y por lo tanto el impacto es menor sobre la velocidad de manejo de la memoria.

También se puede ver que como impacto directo de la traducción de la representación se logra que el rendimiento de las comunicaciones con las máquinas de igual representación de datos sea superior. Dado que siempre el proceso *ping* se asigna a una PC con Linux (**purmamarca**), el rendimiento es superior con todas las demás PCs (**tilcara**, **fourier** y

cf1), comparándolo con la alternativa anterior.

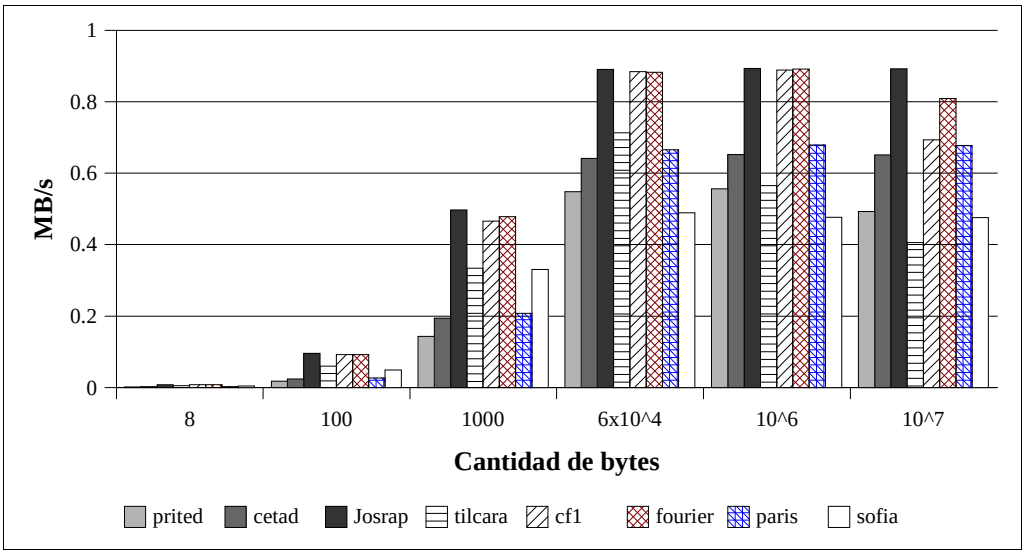


Figura C.9: MB/s con Ruteo *PvmDefault* y Traducción de Representación.

Quizás lo más significativo en ambos casos (codificación XDR y traducción de representación, que es lo único que ha variado hasta ahora) es el muy bajo rendimiento de la computadora **sofia**, que no llega a 0.6 MB/s en ningún caso. Este bajo rendimiento se acentúa cuando se la compara con computadoras varias veces inferiores en velocidad de procesamiento relativa como lo son las computadoras **prited**, **cetad** y **paris**.

C.7.3 Rendimiento con Ruteo entre Tareas de Usuario y con Codificación

La Figura C.10 muestra los tiempos de comunicación obtenidos cuando las tareas de usuario se comunican directamente utilizando el protocolo TCP y con la codificación de datos XDR (PvmDataDefault). Esta alternativa de transmisión de los mensajes es la recomendada por la documentación de PVM cuando las computadoras (y su forma de representar los datos) son heterogéneas.

Comparando los resultados de la Figura C.10 con los anteriores, se pueden observar algunas diferencias menores, como por ejemplo:

- el tiempo de latencia es menor (más cercano a 1 ms que a los 10 ms),
- en escala logarítmica, los tiempos para mensajes pequeños (hasta 1000 bytes) las computadoras no presentan diferencias notables. Nuevamente se debe notar que la escala logarítmica enmascara muchas diferencias que luego se visualizan con claridad en términos de MB/s.

Lo realmente sorprendente que se puede notar en la Figura C.10 es el excesivo tiempo de comunicaciones para mensajes mayores a los 1000 bytes en **prited** y en **sofia**. Aún en la escala logarítmica la diferencia es notable y es muy interesante que, en principio, sucede

con dos computadoras que no están relacionadas (en términos de diseño) de ninguna manera en cuanto a hardware ni a software; **prited** y **sofia**: una Sun SPARCStation 2 de principio de los '90 y una IBM RS/6000 de finales de la misma década.

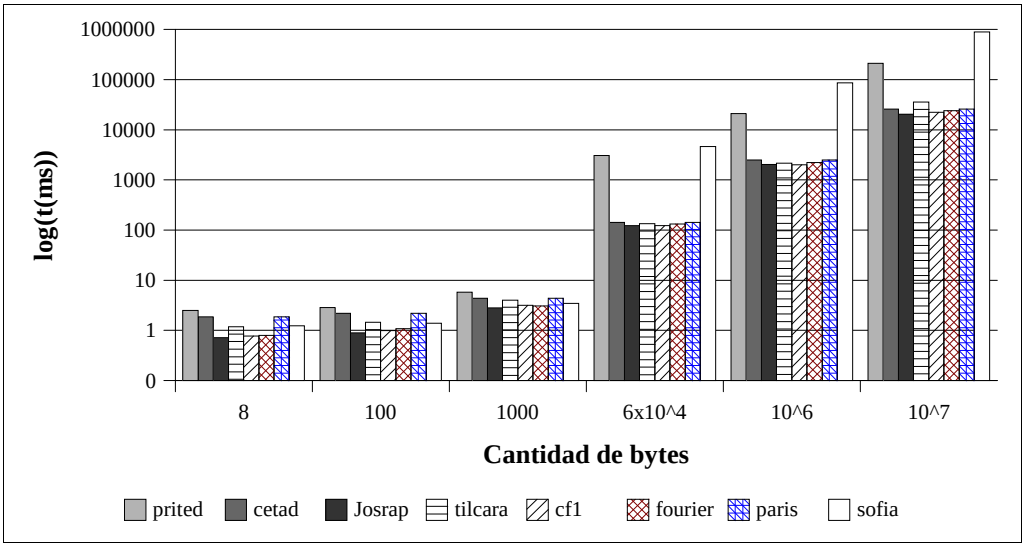


Figura C.10: Tiempos con Ruteo entre Tareas y Codificación *PvmDefault*.

La Figura C.11 muestra los mismos resultados pero en función de MB/s. Ahora las diferencias entre computadoras son más visibles, especialmente con longitud de mensajes mayor a 1000 bytes.

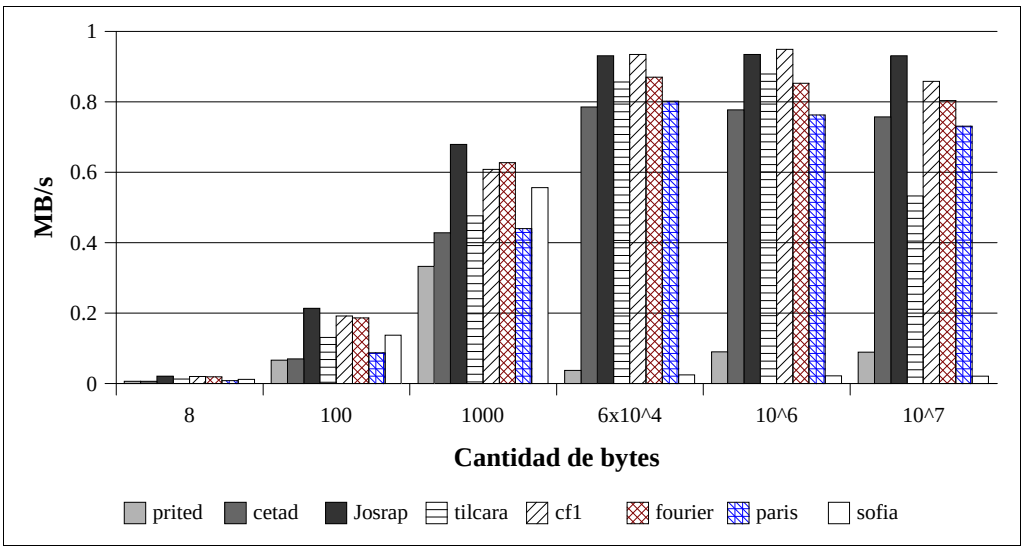


Figura C.11: MB/s con Ruteo entre Tareas y Codificación *PvmDefault*.

La última de las conclusiones enumeradas tiene demasiada importancia dado que se opone totalmente a la documentación y a los reportes al respecto que se han dado de PVM. En este sentido, se verificó la instalación de PVM, los programas *pingpong*, y las posibles razones por las cuales se podría dar esta situación y no se encontró ninguna razón

significativa en relación con la biblioteca PVM.

Pruebas preliminares de conexiones TCP entre las computadoras **purmamarca** y **sofia** mostraron rendimiento similar a nivel de procesos de usuario. Esto implica que las rutinas de comunicaciones de PVM no hacen más que mostrar lo que sucede a nivel de conexión entre las computadoras. La similitud (en términos de rendimiento de las comunicaciones) entres las computadoras **sofia** y **prited** hace asumir el mismo comportamiento a nivel de conexiones TCP.

C.7.4 Rendimiento con Ruteo entre Tareas de Usuario y con Traducción de Representación

La Figura C.12 muestra los tiempos de comunicaciones obtenidos para el ruteo entre tareas (TCP) y con traducción de representación de los datos.

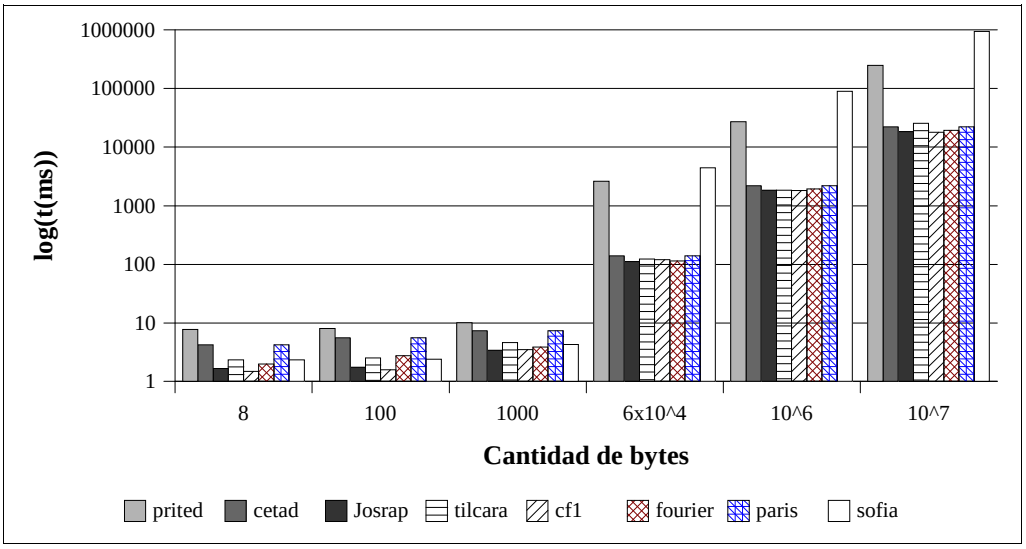


Figura C.12: Tiempos con Ruteo entre Tareas y Traducción de Representación.

La Figura C.13 muestra los mismos resultados en función de MB/s que se obtiene entre tareas de usuario comunicadas con las funciones de PVM, excepto en lo que se refiere a la traducción de la representación de datos, que es propia.

Con esta alternativa se supera por primera vez 1 MB/s de ancho de banda para las comunicaciones entre computadoras con la misma representación de datos. Tanto **purmamarca** donde está ubicada siempre el proceso *ping* como **Josrap**, **tilcara**, **fourier** y **cetadfomec1** (y **cetadfomec2**) son PCs (aunque con distintos procesadores) con sistema operativo Linux. Por lo tanto, los mensajes pueden ser enviados con PvmDataInPlace, lo cual implica que no hay copiado en buffers, ni codificación que pueda incrementar el tamaño final de los datos a transmitir sobre la red de interconexión de computadoras.

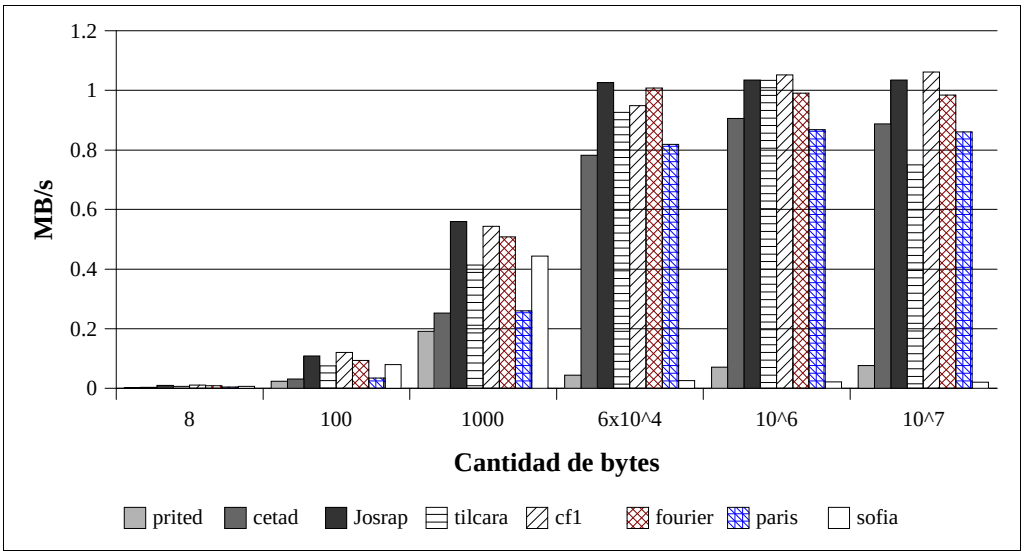


Figura C.13: MB/s con Ruteo entre Tareas y Traducción de Representación.

La información que aportan estas figuras es muy similar a la que aportan las dos anteriores (Figura C.10 y Figura C.11) tanto en valores absolutos como en la relación existente entre las computadoras. También muestra (y en este sentido de alguna manera confirman los resultados anteriores), la gran diferencia en rendimiento que existe entre las computadoras **prited** y **sofia** con respecto a las demás.

El muy bajo rendimiento de **prited** y **sofia** con esta alternativa de comunicaciones (ruteo TCP entre tareas y traducción de representación) entre tareas de PVM hace posible descartar definitivamente que el problema pueda ser por la codificación. Por el contrario, el problema parece ser de las comunicaciones TCP y/o de las comunicaciones de PVM con ruteo TCP.

C.7.5 Conclusiones de la experimentación con PVM

Si se espera utilizar el máximo rendimiento posible de las comunicaciones se debe observar con más cuidado la computadora **sofia**, de la cual es esperable un rendimiento mucho mayor. En el caso particular de los mensajes con ruteo directo, el rendimiento obtenido es realmente muy inferior al esperable y por lo tanto es posible que haya algún error en la forma en que se manejan las comunicaciones TCP en el sistema operativo o en cuanto a la configuración (a nivel de sistema operativo) de las conexiones TCP de algunas computadoras. En este aspecto, la experimentación no hizo más que mostrar que puede haber un (serio) problema en el rendimiento causado por un problema de software.

También en el contexto de rendimiento, es interesante recordar que *todas* las computadoras tienen la misma capacidad en cuanto a hardware de comunicaciones. Esto significa que todas las computadoras, independientemente del fabricante, tienen interfase (NIC) Ethernet 802.3, y por lo tanto todas serían capaces de comunicarse a 10 Mb/s, lo cual implica en teoría 1.25 MB/s (1.25×2^{20} bytes por segundo). Se puede sospechar cierta pérdida de

rendimiento sostenido dada por la sobrecarga (overhead) impuesta por el sistema operativo y sus buffers, procesos, etc. más todo lo que se refiere al mismo PVM, pero no se tiene *a priori* una idea cuantificada de la pérdida de rendimiento que toda esta sobrecarga implica. Por lo tanto, siempre puede quedar pendiente conocer el máximo rendimiento posible de las comunicaciones para las tareas de usuario si solamente se monitoriza el rendimiento con el método de *pingpong* con tareas utilizando PVM.

Volviendo al objetivo inicial de la estimación de α y β, se tienen algunas conclusiones muy importantes: el rango de los valores y la heterogeneidad de los valores.

- Dependiendo de la alternativa de comunicación de mensajes elegida, la latencia varía entre 1 y 10 ms.
- También dependiendo de la alternativa elegida (y excluyendo los casos particulares de **prited** y **sofia**), la tasa de transferencia (o ancho de banda) varía entre algo menos de 0.5 MB/s y algo más de 1 MB/s.
- Sea cual sea la alternativa elegida, tanto la latencia como la tasa de transferencia dependen de la computadora.

Esta última conclusión es un tanto desalentadora en cuanto a que, un subsistema que es en teoría homogéneo como el de la interconexión de computadoras, "se transforma" en heterogéneo cuando se mira desde las tareas de usuario que componen una aplicación paralela.

La consecuencia inmediata de la heterogeneidad de tiempos de comunicación en redes Ethernet con PVM es: un mensaje que debería llegar en la misma cantidad de tiempo a cualquier proceso en otro procesador, ahora llegará en una cantidad de tiempo dependiente de los procesadores origen y destino, más allá de la solución adoptada para resolver las diferencias de representación de datos.

Para intentar corroborar los resultados obtenidos con PVM, se diseña un nuevo conjunto de experimentos que intentan aumentar la precisión en cuanto a:

- α y β para procesos de usuario
- heterogeneidad o no de las comunicaciones
- los casos particulares de bajo rendimiento de **prited** y **sofia**.

C.8 Experimentación con el Comando ping de Linux

El comando *ping* (al menos en su *versión* del sistema operativo Linux) es suficientemente versátil como para:

- Medir tiempos de comunicaciones a nivel de aplicaciones de usuario, y
- Generar "mensajes" de distintas longitudes.

De hecho, cualquier usuario sin permisos especiales puede ejecutar el comando *ping*, lo cual genera un proceso de usuario que produce un *paquete pingpong* por segundo a nivel

de protocolo ICMP (Internet Control Management Protocol). El tiempo de ida y vuelta del paquete es reportado por el mismo comando *ping*, con lo cual no hay que agregar ningún tipo de instrumentación. La longitud del *paquete pingpong* se puede variar utilizando una opción del comando, y esto a su vez permite observar el rendimiento de la red de interconexión en función de la cantidad de datos transferidos, considerando al *paquete pingpong* como un mensaje.

La Figura C.14 muestra los tiempos de comunicación involucrados para distintas cantidades de bytes que contienen los *paquetes pingpong*.

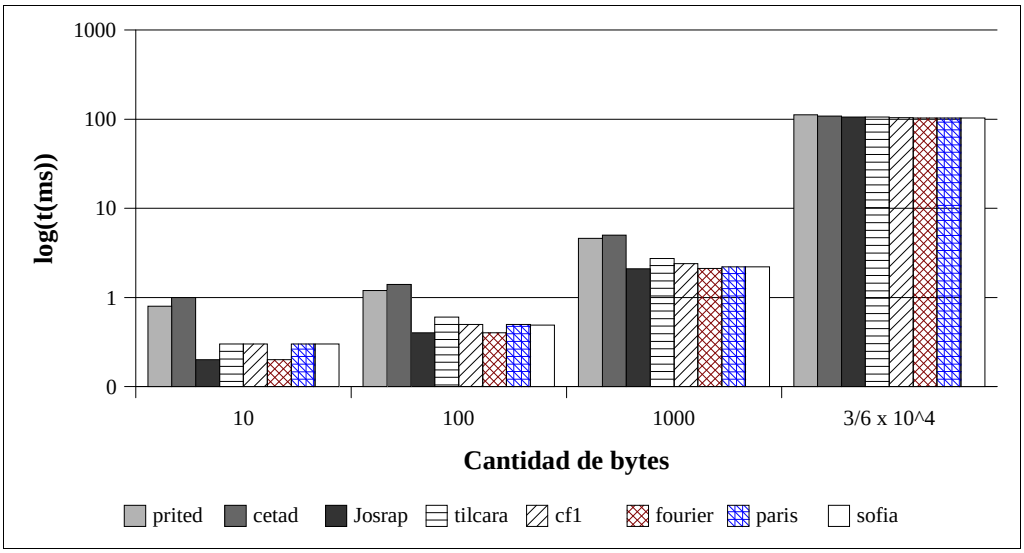


Figura C.14: Tiempos con ping de Linux.

Es necesario hacer algunas aclaraciones con respecto a las longitudes de los *paquetes pingpong* así como también al último identificador que aparece como “3/6 x 10^4”. Todo paquete ICMP debe ser menor a 64 KB tal como los paquetes IP y por lo tanto lo que se puede medir con el comando ping llega a esa cantidad de bytes que se transfieren.

En el caso particular de las computadoras con el sistema operativo Sun 4.1.x (Sun OS basado en BSD), **prited** y **cetad**, por alguna razón no documentada no responden a los paquetes ICMP de más de 32 KB y por lo tanto los pingpong con esas computadoras se llevaron a cabo con tamaño máximo de 30000 bytes (multiplicando luego el tiempo por dos para ecualizarlos con los tiempos de las demás computadoras.

A partir de la información de la Figura C.14 se puede concluir que:

- El tiempo de latencia (α) de mensajes para tareas de usuario (que se comunican con protocolo ICMP) es del orden de 1 ms. Como en PVM, el tiempo de latencia depende de la computadora, aunque el rango de variación es bastante menor.
- Recién a partir de mensajes de 1000 bytes o mayores el tiempo comienza a ser proporcional a la cantidad de datos que se transfieren. Esto significa, como siempre en este contexto, que hasta esa longitud de “mensajes” (en este caso, *paquetes de pingpong*) el tiempo de latencia es suficientemente grande como para ser mayor que el tiempo de transmisión de datos.

La Figura C.15 muestra los mismos resultados pero en términos de MB/s de la transferencia de datos de los paquetes de *pingpong*. Se pueden notar las diferencias entre las computadoras enmascaradas en parte por la escala logarítmica de los tiempos de la figura anterior.

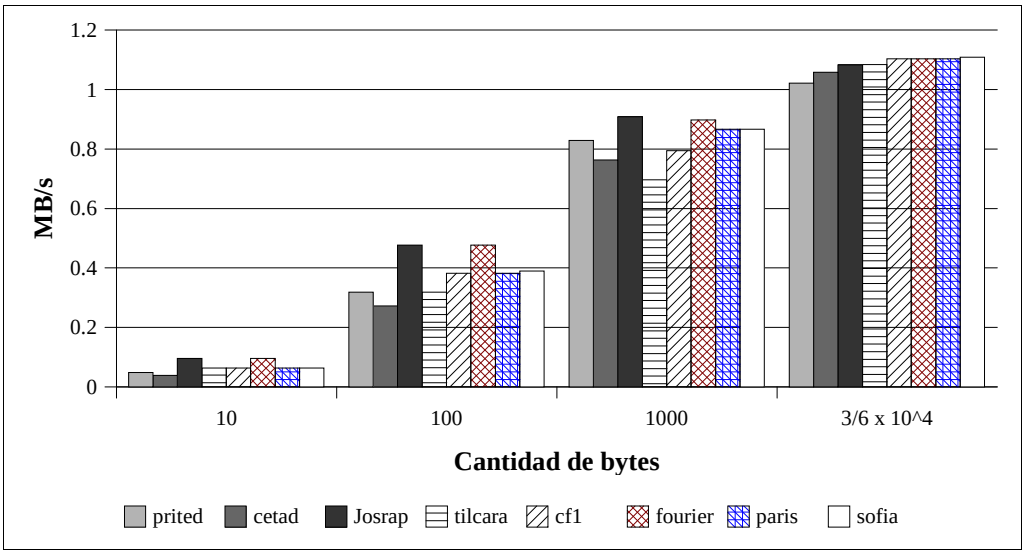


Figura C.15: MB/s con ping de Linux.

Como sucede con la latencia, la tasa de transferencia de información no es exactamente igual para todas las máquinas, pero el rango de variación es mucho menor que el que se tiene cuando los procesos se comunican en PVM. También en términos de tasa de transferencia se confirma que:

- El máximo obtenible con ICMP es casi el físico, con lo cual la diferencia entre este máximo (con el ping de Linux) y el que se obtiene en PVM es debido a la sobrecarga de PVM.
- El caso particular de bajo rendimiento de las computadoras **prited** y **sofia** no se debe a hardware ni a los protocolos más cercanos al hardware (como IP e ICMP).

Como con la experimentación con PVM, recién para tamaños de mensajes del orden de 10^4 bytes (30000 bytes para **prited** y **cetad** y 60000 para el resto) se logra el mejor rendimiento de las comunicaciones. Sin embargo, con estos experimentos se agrega información importante que complementa lo que aportó la experimentación inicial llevada a cabo con PVM.

C.9 Conclusiones de la Experimentación de PVM y ping de Linux

A partir de los resultados obtenidos en PVM y con el comando ping de Linux cabe intentar un intermedio. Es muy probable que no se llegue a tener un rendimiento exactamente igual

al observado con el comando ping porque:
- Con el comando ping no se tienen en cuenta ni se resuelven las diferencias de representación de datos.
- Necesariamente se deben implementar rutinas de comunicación que puedan superar los 64 KB de longitud de mensajes. Expresado de otra manera, lamentablemente no siempre los mensajes a transferir entre procesos se pueden encapsular en un solo paquete de un protocolo de comunicaciones muy cercano al hardware tal como ICMP.

Sin embargo, no parece que estas dos restricciones, aunque fuertes, sean las que hacen que PVM tenga:
- Tan bajo rendimiento de comunicaciones, como en el caso de **prited** y **sofia** con ruteo directo.
- Rendimiento tan heterogéneo dependiente de las computadoras entre las cuales se transfieren datos.

De alguna manera, por un lado el comando ping parece dar una versión muy optimista del rendimiento de la red de comunicaciones y por el otro, PVM parece tener una sobrecarga muy alta. Esta gran sobrecarga de PVM produce una gran reducción del rendimiento de la red de interconexión y esta reducción del rendimiento es proporcional a la velocidad relativa de la computadora. Por lo tanto, el rendimiento de las comunicaciones pasa a ser dependiente de las computadoras que se comunican y tan heterogéneo como la heterogeneidad de las computadoras.

Debido a esto, parece natural buscar una solución intermedia en cuanto a rendimiento de la red de interconexión de computadoras. Esto implica tener e nivel de los procesos de una aplicación paralela de usuario mejor rendimiento y más homogéneo (de acuerdo al hardware de comunicaciones) que el que se obtiene con PVM aunque quizás no sea el obtenido con el comando ping.

Además, es necesario recordar que, si bien toda la experimentación (y por lo tanto las conclusiones que a las que se llegaron) son basadas en el método de *pingpong*, es decir por comunicaciones punto a punto entre dos y sólo dos procesos:
- el problema original a resolver es el de la multiplicación de matrices,
- el algoritmo desarrollado para multiplicar matrices en redes de computadoras está basado en broadcasts,
- es importante tener una implementación eficiente de los broadcasts dada su amplia utilización en los programas paralelos [11].

Como se puede concluir a partir de la experimentación que se mostró en este capítulo, el rendimiento de las comunicaciones punto a punto con PVM no es del todo satisfactorio. Por lo tanto, no hay razón para suponer que el rendimiento de las comunicaciones colectivas, que son las más importantes para la aplicación que se está analizando y para muchas otras, sí sea satisfactorio.

Por las razones expuestas, en el capítulo siguiente se explicará el desarrollo y el rendimiento obtenido en principio para una rutina de comunicaciones colectivas broadcast que puede ser extendida a una biblioteca de comunicaciones colectivas.

C.10 Broadcasts Basados en UDP

Los principales objetivos de rendimiento para el desarrollo de una rutina broadcast distinta de la que proveen bibliotecas de pasaje de mensajes como PVM y las implementaciones de MPI son:

- Mejorar el rendimiento obtenido desde los procesos de usuario en relación al obtenido con las bibliotecas de "propósito general" como PVM.
- Obtener homogeneidad de rendimiento de acuerdo a la homogeneidad del hardware, que en caso de PVM no se verifica en la experimentación.

Además, utilizando PVM, las formas de enviar un mismo mensaje a más de un proceso destino son dos:

- La rutina de multicast, **pvm_mcast().**
- la rutina de broadcast en un grupo, **pvm_bcast().**

Y la implementación de ambas rutinas se basa en múltiples mensajes punto a punto. Es decir que tanto pvm_mcast() como pvm_bcast() implican que, como mínimo, el mismo mensaje es enviado *m* veces desde la computadora origen (donde está ejecutándose el proceso que envía el mensaje) hacia las *m* máquinas donde hay al menos un proceso destino del mensaje. Si, por ejemplo, un mensaje broadcast o multicast tiene cinco receptores y cada uno de estos procesos receptores está ejecutándose en una máquina distinta (y distinta de la máquina en donde se ejecuta el proceso que envía el mensaje), el tiempo total del mensaje será aproximadamente igual a cinco veces el tiempo del mismo mensaje si se envía a otro proceso que se ejecuta en otra máquina. Para estas comunicaciones punto a punto entre máquinas se utilizan las mismas rutinas con las que se ha realizado la experimentación.

En principio, se requiere una rutina para mensajes broadcast con rendimiento *aceptable* en redes Ethernet, donde aceptable se puede definir de acuerdo con:

- Valores absolutos de tiempo de comunicaciones más cercanos a los que provee el hardware de los que se vieron en la experimentación con PVM.
- Escalabilidad en cuanto a cantidad de máquinas, dado que al aprovechar las capacidades de broadcast de las redes Ethernet el mismo mensaje puede ser enviado y recibido hacia/en tantas computadoras como haya conectadas. Evidentemente habrá una penalización dependiente de la cantidad de computadoras que reciben un mismo mensaje por razones de sincronización y mantenimiento de la confiabilidad de los datos que se transfieren, pero esta penalización debería ser muy lejana a la repetición del mismo mensaje tantas veces como computadoras distintas deben recibirlo.

El rendimiento de pvm_mcast() como pvm_bcast() no es aceptable y por lo tanto la biblioteca PVM ya no sería útil para los mensajes broadcast que requiere el algoritmo de multiplicación de matrices en paralelo. En este punto se tienen varias alternativas, siendo las dos más importantes:

1. Utilizar otra biblioteca de pasaje de mensajes, tal como alguna implementación de MPI, que es lo que se suele recomendar para este tipo de arquitecturas paralelas.
2. Implementar una rutina de mensajes broadcast (y eventualmente toda una biblioteca de

comunicaciones colectivas) para utilizar explícitamente la capacidad de broadcast de las redes Ethernet.

La utilización de otra biblioteca de pasaje de mensajes tiene, en principio un inconveniente fundamental desde el punto de vista de rendimiento, o de "predicción" de buen rendimiento de los mensajes broadcast. En el caso específico de MPI, es claro que el rendimiento es dependiente de la implementación. Más específicamente, la implementación será la que determina el grado de aprovechamiento de la/s característica/s de las redes Ethernet para los mensajes broadcast. En este sentido, MPI y en particular todas sus implementaciones comparten *cierto grado de incertidumbre* respecto del rendimiento de los mensajes broadcast con todas las demás bibliotecas de pasaje de mensajes, incluida PVM. La diferencia en este caso son los experimentos específicos se llevaron a cabo y que determinaron las características de rendimiento de los mensajes broadcast (y multicast) para PVM y no para las demás bibliotecas. De hecho, es bastante difícil que las bibliotecas de pasaje de mensajes sean optimizadas para las características de las redes Ethernet dado que:

- En general, las bibliotecas son propuestas de una forma u otra como estándares para las máquinas paralelas de pasaje de mensajes y por lo tanto no tiene sentido orientarlas hacia un tipo específico de redes de interconexión. De hecho, tanto PVM como MPI se han implementado para distintos tipos de máquinas paralelas y por lo tanto no tiene sentido orientarlas *a priori* a las redes de interconexión Ethernet.
- En general, las bibliotecas proveen una gran cantidad de rutinas de comunicación. Aunque en teoría se puede afirmar que con las primitivas **send** - **receive** para la comunicación de procesos punto a punto es suficiente, también se ha concluido que existe una gran variedad de rutinas de comunicación que se consideran útiles y hasta necesarias en algunos casos. Quizás el ejemplo más claro al respecto sea la propia definición del estándar MPI. En este contexto, es muy difícil orientar u optimizar una o una clase de rutinas de comunicaciones para una o una clase de redes de interconexión sin producir una biblioteca excesivamente costosa (en términos de desarrollo, mantenimiento, etc.) y/o *demasiado* específica.

Por estas razones, se eligió implementar una rutina de mensajes broadcast entre procesos de usuario con un conjunto de premisas de diseño e implementación, de forma tal que:

- Aproveche el *propio* broadcast de las redes Ethernet, y de esta manera se optimiza en cuanto a rendimiento. Dado que el algoritmo depende exclusivamente de los mensajes broadcast, al aprovechar el broadcast de las redes Ethernet automáticamente se tiene una buena expectativa en cuanto a escalabilidad ya que se espera que el tiempo de comunicaciones se mantenga constante y no aumente de manera proporcional a la cantidad de computadoras que se utilizan.
- Sea suficientemente simple para no imponer una carga demasiado pesada en cuanto a implementación y mantenimiento. Además, es claro que simplicidad *per se* hace un gran aporte para el rendimiento óptimo. Por otro lado, la propuesta es suficientemente específica como para que la implementación sea simple.
- Con el máximo de portabilidad posible, para ser utilizada si es posible incluso en el contexto de las redes de interconexión que no sean Ethernet.
- Implementada e instalada desde modo usuario, sin cambiar el sistema operativo (el *kernel*) y sin que sea necesario obtener permisos especiales (*superuser*). Es normalmente aceptado que los mejores resultados en cuanto a rendimiento se obtienen

adaptando el kernel y/o con la posibilidad de manejar las prioridades de los procesos, tal como en [5] [4] [12]. Estas posibilidades se descartan dado que:
    - En general, las bibliotecas de uso libre no utilizan estas características y por lo tanto sería como cambiar el contexto de desarrollo de software paralelo. Básicamente, un usuario que haya utilizado siempre PVM nunca tuvo ni tiene por qué obtener prioridades especiales y aún cambiar el mismo sistema operativo.
    - La propuesta original se dirige a redes de computadoras instaladas y por lo tanto cada computadora no necesariamente tiene como objetivo único y/o principal el cómputo paralelo. De hecho, se puede tener el caso de distintos administradores para cada una de las computadoras a utilizar en paralelo y esto produce, por lo menos, un múltiple trabajo de administración que en general no es fácil de resolver.
- Eventualmente pueda ser extendida a toda una biblioteca de comunicaciones colectivas, tal como otras propuestas [2] [1] [3] pero orientada específicamente a las redes de interconexión Ethernet.

La mayoría (sino *todas*) las premisas anteriores se cumplen al basar todo el diseño y la implementación de la rutina de broadcast en el protocolo estándar UDP (User Datagram Protocol) [8] sobre IP (Internet Protocol) [9] ya que:
- UDP permite enviar un mismo dato o conjunto (paquete) de datos a múltiples destinos a nivel de aplicaciones de usuario.
- Tal como se verificó en todas las máquinas utilizadas, la implementación del protocolo UDP aprovecha directamente la capacidad de broadcast de las redes Ethernet.
- En principio, parece razonable que el broadcast implementado directamente como parte del protocolo UDP tenga mejor rendimiento que el implementado por un usuario. Si en una red ATM, por ejemplo, se tiene la posibilidad de utilizar UDP es muy probable que el broadcast de UDP sea *mejor* (en términos de rendimiento) que el que se pueda implementar desde los procesos de usuario. Aunque el rendimiento no se tenga en cuanta, siempre que exista una implementación del protocolo UDP sea podrá utilizar el broadcast propuesto, independientemente de que la red de interconexión sea Ethernet o no.
- La interfase de usuario provista por los sockets es suficientemente simple y ampliamente extendida a todas las versiones de UNIX, como para simplificar la implementación de la rutina de broadcast, aún cuando se deben resolver problemas relacionados con *sincronización* de procesos (en una misma y en diferentes computadoras) y *confiabilidad* de las comunicaciones.
- Los protocolos UDP, TCP e IP son fácilmente utilizables desde los procesos de usuario, al menos en las computadoras estándares de las redes locales instaladas.

En resumen, se propone una *nueva* rutina de mensajes broadcast basada en UDP y portable como mínimo a todas las versiones de UNIX utilizadas en todas las redes locales en las cuales se lleva a cabo la experimentación. Con esta rutina de mensajes broadcast se realizaron los mismos experimentos que para las rutinas de comunicación de PVM. Inicialmente se muestran los resultados de las comunicaciones punto a punto (mensaje “broadcast” o “multicast” utilizando un solo proceso receptor) y por último también se muestran los resultados de mensajes broadcast con pvm_mcast() y pvm_bcast() de PVM y con la rutina propuesta basada en UDP.

C.10.1 Un Unico Receptor (Mensajes Punto a Punto)

Con el fin de comparar los resultados con los que ya se obtuvieron por la experimentación con las rutinas de comunicaciones punto a punto de PVM y con el comando ping de Linux se utilizó la rutina de mensajes broadcast con un único proceso receptor. Los resultados referentes al tiempo de comunicaciones se muestran en la Figura C.16. Se debe destacar que estos tiempos de comunicaciones entre dos máquinas pueden no ser óptimos dado que la rutina de comunicaciones está diseñada para mensajes broadcast que involucran más de una computadora que debe recibir mensajes. De todas maneras se puede utilizar para comparar con los resultados que se muestran en

- La Figura C.14 con el rendimiento de las comunicaciones de acuerdo con el comando ping (protocolo ICMP).
- La Figura C.12 con el rendimiento de las comunicaciones de PVM con ruteo entre tareas (protocolo TCP) y traducción de la representación de los datos.
- La Figura C.8 con el rendimiento de las comunicaciones de PVM con ruteo entre los procesos de PVM (pvmd, protocolo UDP) y traducción de la representación de los datos.

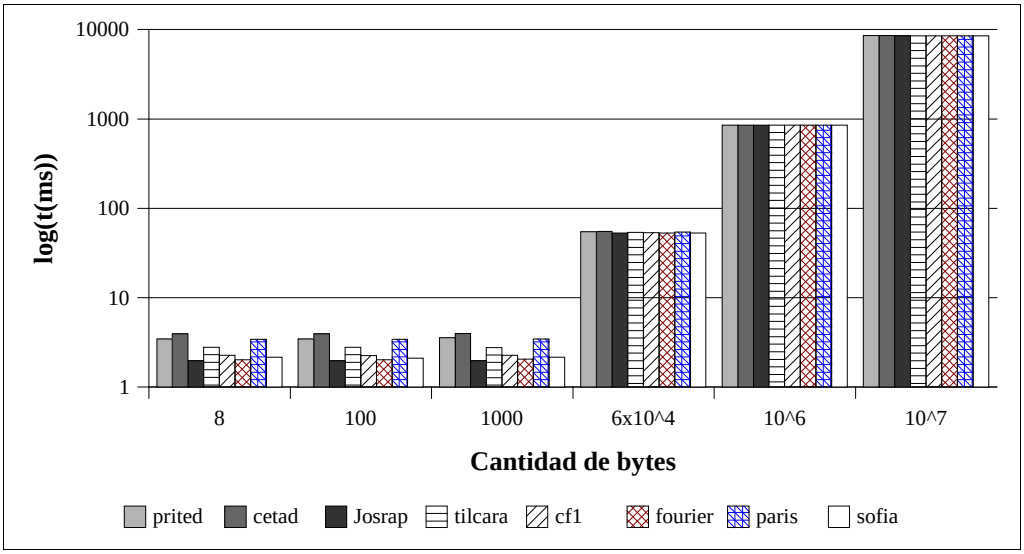


Figura C.16: Tiempos “Punto a Punto” con Broadcast Basado en UDP.

Comparando estos resultados con los del comando ping de Linux (Figura C.14) se tiene que:

- La latencia es bastante mayor y se verifica que al menos hasta la transferencia de mil bytes el tiempo de latencia es el que domina el tiempo total. Por otro lado, se confirma que la latencia de las comunicaciones es dependiente de las máquinas involucradas en la transferencia de datos.
- El tiempo de comunicaciones es similar al que se obtiene con el comando ping de Linux para 30000 (estaciones de trabajo **cetad** y **prited**) y 60000 bytes (las demás computadoras).

Comparando estos resultados con los de las rutinas de comunicaciones punto a punto de PVM (Figura C.8 y Figura C.12) se tiene que:

- La latencia es similar a la que se tiene con PVM, y en muchos casos casi idéntica.
- Las computadoras **prited** y **sofia** no muestran ninguna anomalía de rendimiento para ningún tamaño de mensajes. Esto confirma que el problema se debe a la configuración y/o a la implementación del protocolo TCP en esas computadoras.
- Al menos a partir de la longitud de mensajes de 60000 bytes, el rendimiento de las comunicaciones es homogéneo, tal como es de esperar desde el punto de vista del hardware de comunicaciones.
- El rendimiento con el broadcast basado en UDP es bastante superior al que se tiene con las rutinas de comunicaciones punto a punto de PVM. Aún a pesar de la escala logarítmica, se puede notar que el tiempo de comunicaciones es muy cercano al óptimo, tal como si no hubiera sobrecarga (overhead) de las distintas capas de software involucrado (sistema operativo, la rutina de broadcast, etc.). Por ejemplo, el tiempo de comunicaciones de los mensajes de 10^6 bytes es muy cercano a un segundo (o 1000 milisegundos, tal como se muestra en la Figura C.16).

La Figura C.17 muestra los mismos resultados en términos de ancho de banda asintótico, donde se puede verificar con claridad que los resultados son altamente satisfactorios en términos de rendimiento.

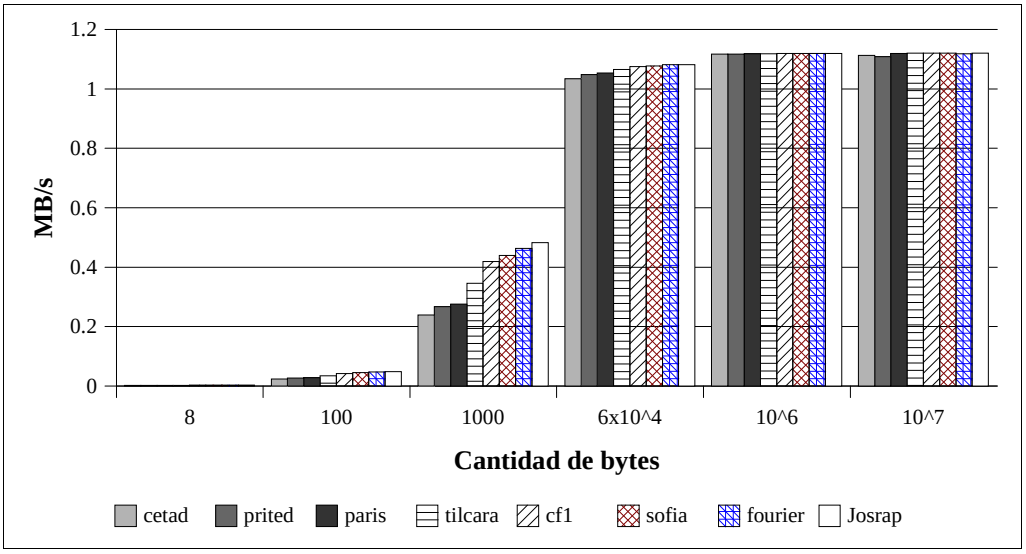


Figura C.17: MB/s “Punto a Punto” con Broadcast Basado en UDP.

Con estos resultados de comunicaciones punto a punto se tiene, como mínimo:

- Casi todo el rendimiento del hardware de comunicaciones a nivel de los procesos de las aplicaciones paralelas. La sobrecarga de todas las capas intermedias de software casi no afecta el rendimiento final entre los procesos.
- En ausencia de colisiones, el rendimiento es independiente de la red de comunicaciones es independiente de las computadoras involucradas. La heterogeneidad de las computadoras con sus diferencias relativas en cuanto a capacidad de cálculo no se traduce, como en PVM, a “rendimiento heterogéneo”.

C.10.2 Mensajes Broadcasts

El objetivo final de la rutina de comunicaciones broadcast no es la transferencia de datos de un proceso a otro, sino la transferencia de un proceso a una cantidad determinada de procesos que se ejecutan en diferentes computadoras. Es por esta razón que se debe verificar al menos por experimentación que se pueden llevar a cabo mensajes broadcast entre procesos con rendimiento cercano al óptimo y relativamente independiente de la cantidad de procesos receptores, o computadoras involucradas en los mensajes broadcast.

La Figura C.18 muestra los tiempos de comunicaciones involucrados para distintas longitudes de mensajes broadcast y diferentes cantidades de receptores asignados en diferentes máquinas. Con el objetivo de comparar mejor las diferentes longitudes de mensajes se elige mostrarlas a todas en mismo gráfico en vez de mostrar un gráfico por cada longitud de mensajes. Separadas por las líneas de puntos verticales, sobre el eje **x** se muestran los tiempos de mensajes broadcast para distinta cantidad de receptores (máquinas). Los resultados que se muestran de PVM son independientes de la utilización de la rutina pvm_mcast() o pvm_bcast(), porque tienen rendimiento similar.

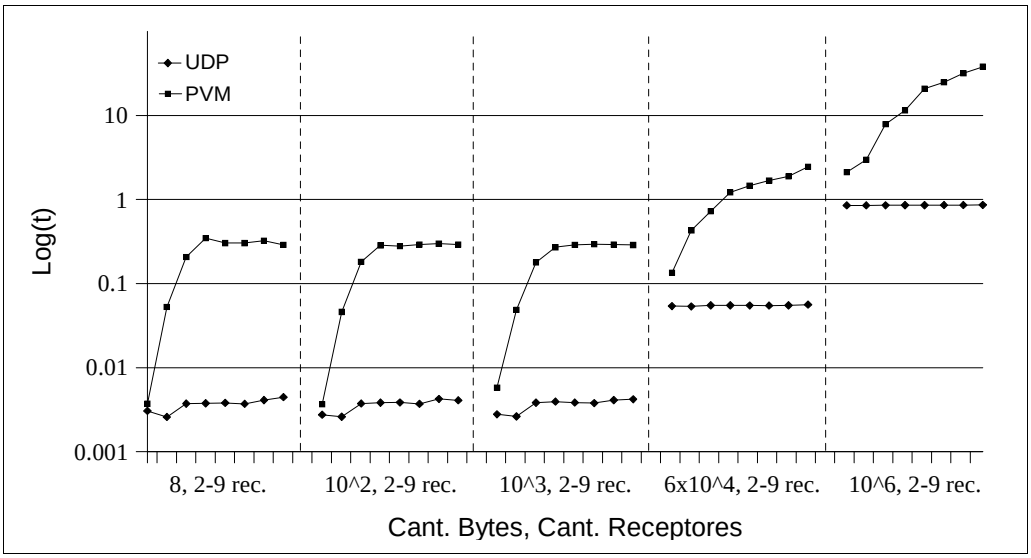


Figura C.18: Tiempos de Broadcast con PVM y Basado en UDP.

Se puede notar que el tiempo con las rutinas de PVM se mantiene bastante independiente de la longitud de mensajes de 8, 100 y 1000 bytes respectivamente. De hecho, depende más de las máquinas involucradas. Para estas longitudes de mensajes la rutina propuesta que se basa en UDP utiliza tiempos relativamente constantes y también independientes de la cantidad de computadoras involucradas. Para los mensajes broadcast de 60000 y 1000000 de bytes el tiempo es directamente proporcional a la cantidad de receptores cuando se utilizan las rutinas de PVM. Una vez más, cuando se utiliza la rutina basada directamente en UDP el tiempo (además de ser un poco mejor que el mejor de PVM) es relativamente constante e independiente de las computadoras involucradas.

La Figura C.19 muestra los mismos resultados pero en términos de MB/s y por lo tanto se pueden notar algunos detalles enmascarados por la escala logarítmica de tiempos del gráfico anterior. Para el cálculo del ancho de banda asintótico o tasa de transferencia o MB/s de un mensaje broadcast se debe recordar que el mensaje es único, los mismos datos deben llegar a múltiples destinos independientemente de que la implementación se haga con múltiples mensajes punto a punto o de otra manera.

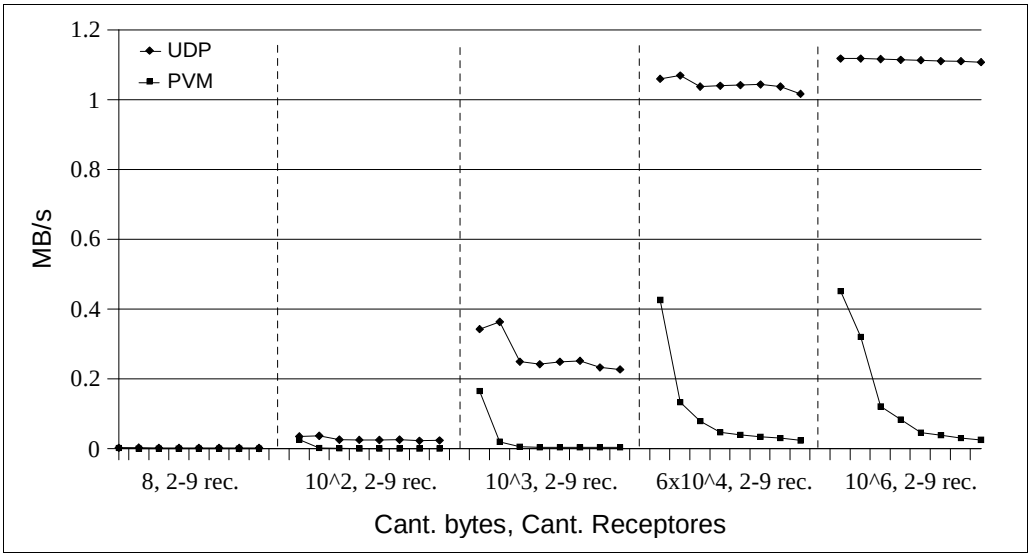


Figura C.19: MB/s de Broadcast con PVM y Basado en UDP.

Como se explicó anteriormente, el tiempo de comunicaciones es dominado por la latencia al menos para los mensajes de hasta 1000 bytes. Por lo tanto, el rendimiento en términos de ancho de banda o tasa de transferencia es bastante pobre utilizando PVM o la rutina de broadcasts basada en UDP. Para los mensajes de 60000 bytes se puede notar fácilmente que el rendimiento al utilizar las rutinas provistas por PVM es dependiente de la cantidad de receptores y es relativamente constante cuando se utiliza la rutina basada en UDP. Las variaciones en este último caso se deben básicamente a que para esta longitud de mensajes la latencia de cada máquina se hace notar en el tiempo total de las comunicaciones. Para los mensajes de 10^6 bytes la latencia de cada máquina ya es mucho menor al tiempo de transferencia de los datos y por lo tanto el rendimiento tiene mucha menor variación en ancho de banda asintótico. En el caso de PVM, se verifica una vez más que el rendimiento disminuye a medida que la cantidad de computadoras involucradas aumenta y por lo tanto se puede afirmar que la implementación del broadcast utiliza múltiples mensajes punto a punto.

C.11 Broadcasts en la Red Local del LQT

La experimentación en la red local del LQT es similar en cuanto a los mensajes broadcast. Se dan las mismas características de rendimiento dependiente de la cantidad de

computadoras involucradas para las rutinas de PVM y rendimiento casi constante para la rutina propuesta basada directamente en UDP. A diferencia de lo que sucede en el CeTAD la heterogeneidad es menor ya que, por ejemplo, no hay diferencias en cuanto a la representación de datos, dado que todas las computadoras disponibles son PC. También a diferencia de lo que sucede en el CeTAD la cantidad de computadoras es menor y por lo tanto la máxima degradación de rendimiento (cuando se utilizan todas las computadoras) es también menor.

Una de las diferencias más notables que arroja la experimentación del LQT con respecto a la detallada del CeTAD se refiere a la latencia de los mensajes punto a punto. Dado que las máquinas son bastante más similares y las diferencias de rendimiento en cuanto a capacidad de cómputo son más reducidas, la latencia de los mensajes está mucho más acotada en cuanto al rango de valores absolutos. De hecho, esto independiza aún más el rendimiento de las comunicaciones entre procesos de usuario dado que ahora no solamente el ancho de banda asintótico es cercano al del hardware de la red de interconexión (utilizando la rutina de mensajes broadcast basada en UDP) sino que el tiempo completo de comunicaciones es mucho más cercano al que se puede estimar con los valores de hardware.

C.12 Broadcasts en la Red Local del LIDI

Como es de esperar en la red local del LIDI, el rendimiento de las comunicaciones es mejor dado que la red de comunicaciones es Ethernet de 100 Mb/s en vez de 10 Mb/s como las del CeTAD y del LQT. Aún así se confirma con la experimentación que para PVM el tiempo de los mensajes broadcast es dependiente de la cantidad de computadoras involucradas. También se confirma que el tiempo de los mensajes broadcast es relativamente independiente de la cantidad de máquinas involucradas cuando se utiliza la rutina basada directamente en UDP.

Es muy interesante el impacto de la latencia en el tiempo total de comunicaciones cuanto el ancho de banda del hardware es diez veces mayor. Dado que gran parte de la latencia se debe a la sobrecarga de factores externos a la propia red de interconexión Ethernet (sistema operativo, protocolos, rutinas de comunicaciones entre procesos, etc.) el tiempo absoluto de la latencia es bastante independiente de la capacidad en ancho de banda. Al multiplicar por diez el ancho de banda de la red Ethernet, implícitamente se multiplica (aunque no necesariamente por diez) el “peso” del tiempo de la latencia en el tiempo total de las comunicaciones. La cuantificación de estos factores debería calcularse al menos con el diseño de experimentos específicos que están fuera del alcance de esta Apéndice .

El impacto de la latencia en el tiempo final de las comunicaciones es definitivamente importante teniendo en cuenta la tendencia a utilizar redes con mayor capacidad de ancho de banda. De todas maneras, es de esperar que la latencia a nivel de hardware de interconexión se reduzca proporcionalmente con el aumento de ancho de banda y también que el overhead impuesto para los procesos de usuario se reduzca.

Referencias

[1] Bala V., J. Bruck, R. Cypher, P. Elustondo, A. Ho, C. Ho, S. Kipnis, M. Snir, “CCL: A Portable and Tunable Collective Communication Library for Scalable Parallel Computing”, Proc. of the 8th International Conference on Parallel Processing, IEEE, April 1994.

[2] Banikazemi M., V. Moorthy, D. Panda, “Efficient Collective Communication on Heterogeneous Networks of Workstations”, Proc. International Conference on Parallel Processing, pp. 460-467, 1998.

[3] Barnett M., S. Gupta, D. Payne, L. Shuler, R. van de Geijn, J. Watts, “Interprocessor Collective Communication Library (InterCom)”, Proc. of the Scalable High-Performance Computing Conference '94, Knoxville, TN, USA, IEEE Computer Society Press, pp. 357-364, May 1994.

[4] Chiola G., G. Ciaccio, “Lightweigth Messaging Systems”, in R. Buyya Ed., High Performance Cluster Computing: Architectures and Systems, Vol. 1, Prentice-Hall, Upper Saddle River, NJ, USA, pp. 246-269, 1999.

[5] Ciaccio G., “Optimal Communication Performance on Fast Ethernet with GAMMA”, Proceedings Workshop PC-NOW, IPPS/SPDP'98, Orlando, FL, LNCS No. 1388, Springer, pp. 534-548, April 1998.

[6] Institute of Electrical and Electronics Engineers, IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1984, 1984.

[7] Pacheco P., Parallel Programming with MPI, Morgan Kaufmann, San Francisco, California, 1997.

[8] Postel J., “User Datagram Protocol”, RFC 768, USC/Information Sciences Institute, Aug. 1980.

[9] Postel J., “Internet Protocol”, RFC 791, USC/Information Sciences Institute, Sep. 1981.

[10] Sun Microsystems, Inc. XDR: External Data Representation Standard. RFC 1014, Sun Microsystems, Inc., June 1987.

[11] Wilkinson B., Allen M., Parallel Programming: Techniques and Applications Using Networking Workstations, Prentice-Hall, Inc., 1999.

[12] GAMMA Home Page http://www.disi.unige.it/project/gamma/