Cómputo Paralelo en Clusters
Herramienta de Evaluación de Rendimiento de las Comunicaciones

Fernando G. Tinetti
fernando@ada.info.unlp.edu.ar

Andrés Barbieri
barbieri@lidi.info.unlp.edu.ar

LIDI[1] - CeTAD[2]

Resumen

Las redes de computadoras utilizadas para cómputo paralelo (*clusters*) se están aplicando de manera satisfactoria en múltiples áreas. Uno de los grandes problemas que deben enfrentar las aplicaciones paralelas en este entorno es el del rendimiento de las comunicaciones, dado que el tiempo de comunicación es muy grande con respecto a la cantidad de operaciones que se pueden llevar a cabo de manera local en cualquier computadora. Por otro lado, tanto la topología de la red local, como las herramientas de programación por pasaje de mensajes que se disponen, tienen su propia sobrecarga de comunicaciones que no siempre es sencilla de evaluar y/o predecible *a priori*. En este artículo se presenta una herramienta que estima el rendimiento de las comunicaciones entre procesos de una aplicación paralela utilizando las bibliotecas de pasaje de mensajes más conocidas y de uso libre. En particular, la atención se centra en las comunicaciones punto a punto (un proceso se comunica con otro, básicamente utilizando las primitivas *send-receive*) y en una de las comunicaciones colectivas más utilizadas, el mensaje *broadcast*, que implica enviar datos desde un proceso a *n* otros procesos de la aplicación paralela.

Palabras Clave: Cómputo Paralelo, Clusters de Computadoras, Pasaje de Mensajes, Evaluación de Rendimiento Paralelo, Evaluación de Sobrecarga de Comunicaciones.

1.- Introducción

El avance en el rendimiento secuencial de las computadoras de escritorio actuales las ha convertida en particularmente útiles para ser utilizadas en cómputo paralelo, dado que es muy sencillo su interconexión en redes locales. Desde el punto de vista del hardware de procesamiento estas plataformas pertenecen a la clase MIMD (Multiple Instruction - Multiple Data stream) [12] y el modelo de programación que se ha utilizado extensivamente es el de pasaje de mensajes, derivado de CSP (Communicating Sequential Processes) [14]. En este sentido, las bibliotecas de uso libre de pasaje de mensajes para este tipo de procesamiento paralelo-distribuido tales como PVM [11] e implementaciones de MPI [19] como LAM-MPI [24] y MPICH [25] han sido parte activa en todo el desarrollo y utilización satisfactoria de las redes de computadoras como máquinas paralelas.

Existen numerosas publicaciones con respecto a la utilización de clusters como arquitecturas paralelas, inicialmente con máquinas homogéneas, tales como Beowulf [5], NOW (Network of Workstations) [1] y *clusters* en general [2]. Más recientemente, se han publicado también numerosos reportes en el campo de los clusters de máquinas heterogéneas [3] [4] [18]. Existe cierto consenso con respecto a que la parte que mayor atención requiere tanto en lo referente a paralelización de las aplicaciones como al hardware utilizado y al rendimiento en general es la de las comunicaciones [4]. En esta área, los esfuerzos se han dirigido en tres direcciones principales:

1 Laboratorio de Investigación y Desarrollo en Informática, Fac. de Informática, Calle 50 y 115, 1900 La Plata
2 Centro de Técnicas Analógico-Digitales, Fac. de Ingeniería, Calle 48 y 116, 1900 La Plata

1. Hardware de interconexión con mayor rendimiento que el estándar. En este contexto se incluyen propuestas como PAPERS (Purdue's Adapter for Parallel Execution and Rapid Synchronization) [8] [9] [10] y *channel bonding* en el ámbito de las instalaciones Beowulf [7] [21].
2. Paralelización orientada a la utilización optimizada de las redes de interconexión a través de las rutinas incluidas en las bibliotecas de pasaje de mensajes [6] [4] [3].
3. Reportes de análisis de rendimiento de una biblioteca de pasaje de mensajes en particular o comparaciones específicas de rutinas de comunicaciones de diferentes bibliotecas de pasaje de mensajes de uso libre [MPIBench] [PVM-MPI-comm].

En el contexto de la última de estas líneas de investigación los reportes se han restringido a análisis particulares sobre alguna combinación de (*biblioteca*, *rutina de comunicaciones*, *tipo de red*) donde: b*iblioteca* se refiere a PVM y/o alguna implementación de MPI, *rutina de comunicaciones* se refiere usualmente a las funciones básicas de pasaje de mensajes *send-receive* y *tipo de red* se refiere al hardware de interconexión, normalmente Ethernet 10 Mb/s o 10/100 Mb/s [17]. En la mayoría de los casos (sino en *todos*) es muy difícil la derivación de conclusiones generales y/o la aplicación de una regla de comparación a todos los casos posibles dada la cantidad muy grande de combinaciones posibles de estas tres características y la inclusión de otra/s.

Por lo tanto, se llega a que, aunque el rendimiento de las comunicaciones entre procesos de una aplicación paralela sobre clusters de computadoras es parte fundamental del rendimiento y de la utilización satisfactoria del cómputo paralelo-distribuido aún es muy difícil estimar y/o evaluar el rendimiento del *middleware* (bibliotecas de pasaje de mensajes) que se utiliza. Parte de esta evaluación incluye también el rendimiento que una aplicación de usuario puede obtener de las comunicaciones tal como se disponen desde el sistema operativo. En este sentido, se pueden identificar diferentes capas o niveles de abstracción y/o sobrecarga impuesta a las comunicaciones entre procesos de usuario, tal como lo muestra a modo de ejemplo la Fig. 1.

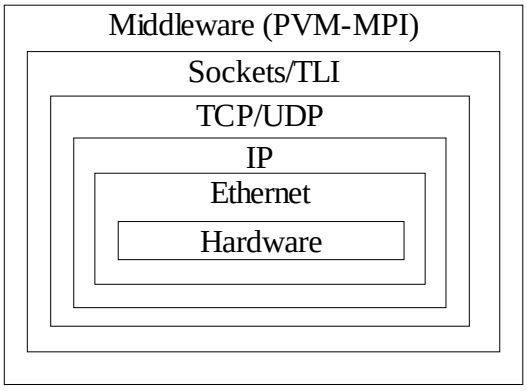

Figura 1: Niveles de Abstracción - Detalle - Sobrecarga de las Comunicaciones.

Si bien es aceptado que las rutinas provistas por la biblioteca que se utiliza para las comunicaciones entre procesos de las aplicaciones paralelas (*middleware*) aportan su penalización en cuanto a rendimiento es muy difícil no solamente su cuantificación sino también la relación de pérdida de rendimiento con respecto al rendimiento del hardware (Ethernet de 10 ó 100 Mb/s, por ejemplo). Es por esto que se propone en este artículo una herramienta de evaluación del rendimiento de las comunicaciones que aportan bibliotecas tales como PVM y algunas implementaciones libres de MPI con el fin de proveer al programador de aplicaciones paralelas sobre clusters un conjunto de parámetros para la elección del middleware más apropiado (en cuanto a rendimiento) como así también una estimación de la sobrecarga que estas bibliotecas imponen sobre la red física de comunicaciones. Se incluyen también como parte de la herramienta posibilidades de evaluación de la comunicación entre procesos de usuario a nivel de las posibilidades que brinda el sistema operativo y que son de menor nivel de abstracción que PVM o MPI.

La sección siguiente se encarga de identificar los modelos que son aceptados para la evaluación de las comunicaciones punto a punto como así también el modelo de evaluación de rendimiento de las comunicaciones colectivas o al menos de las comunicaciones broadcast-multicast que se considera representativa de las comunicaciones colectivas. La tercera sección describe las particularidades de la herramienta que se propone, así como también las principales elecciones de diseño y los problemas y soluciones propuestas en el caso de los clusters heterogéneos, que aportan su propia complejidad. La cuarta sección presenta dos casos de evaluación de las comunicaciones en redes instaladas en las cuales de manera automática se derivan los principales índices de rendimiento al utilizar las bibliotecas de pasaje de mensaje y también los índices calculados a nivel del sistema operativo. Esto permite al usuario derivar casi automáticamente la sobrecarga de las bibliotecas en cuanto a rendimiento de las comunicaciones entre procesos de usuario (pertenecientes a una aplicación paralela). Finalmente se dan las conclusiones y las extensiones inmediatas a la herramienta que se propone en este artículo.

2.- Evaluación del Rendimiento de las Comunicaciones

En general, existe un gran consenso en la literatura en lo referente a la evaluación y/o los índices de rendimiento a tener en cuenta o evaluar para las comunicaciones punto a punto, básicamente cuando se utilizan las primitivas de comunicación *send-receive*. Sin embargo, la situación no es tan clara en el caso de las comunicaciones colectivas que involucran a más de dos procesos y que tienen una gran variedad de opciones y operaciones que se incluyen (*broadcast*, *multicast*, *scatter*, *gather*, *reduce*, etc.).

2.1 Comunicaciones Punto a Punto

Como se aclaró previamente, existe un gran consenso en la literatura respecto a los índices de rendimiento y su estimación para las comunicaciones punto a punto [16] [15] [20] [13] [23]. Se consideran que los dos factores más importantes que determinan el tiempo de comunicación de las transmisiones entre dos computadoras o procesadores de una máquina paralela son básicamente dos:

- Latencia (o *startup time*): es el mínimo tiempo requerido para transmitir cualquier información, incluyendo cualquier overhead del software asociado a las primitivas send-receive. Se suele denotar con el símbolo α.
- Ancho de Banda (o Ancho de Banda Asintótico): se define como la máxima cantidad de datos que pueden ser transmitidos por unidad de tiempo. Se suele denotar con el símbolo β.

Por un lado, el ancho de banda determina el mínimo tiempo de transmisión que se puede esperar por unidad de información. En este sentido, el ancho de banda indica o determina el rendimiento óptimo de las comunicaciones entre procesos. Por otro lado, la latencia indica un mínimo de tiempo que toda comunicación necesariamente incluye. La latencia es un parámetro que se torna relativamente importante en el contexto de los clusters utilizados para cómputo paralelo dado que es varios órdenes de magnitud mayor que en el contexto de las computadoras paralelas *tradicionales*. De hecho, la latencia determina casi de manera unívoca la granularidad mínima de comunicaciones.

Con estos índices α y β (latencia y ancho de banda asintótico) se calcula de manera directa la estimación del tiempo de transferencia de n unidades de información (*bytes*, *floats*, etc.) entre dos procesos como

$$t(n) = \alpha + n/\beta \qquad (1)$$

En el contexto de las computadoras paralelas con arquitectura de memoria distribuida (tal como pueden considerarse los clusters), 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 *ping-pong*. 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 Fig. 2:

Figura 2: Procesos *Ping-Pong*.

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,

En el procesador P_1 se conoce el tiempo total utilizado para la comunicación de los dos mensajes (envío y recepción) 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 la independencia de la sincronización 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.

2.2 Comunicaciones Colectivas

Como se aclara antes, no existe un consenso completo en cuanto a la modelización del rendimiento de las comunicaciones colectivas, y de hecho hay varias propuestas. Las comunicaciones colectivas tienen dos características fundamentales que dificultan la modelización del rendimiento:

1. Involucran a más de dos procesos, y en general a una cantidad *a priori* indeterminada de procesos. Los mensajes de tipo *broadcast* y *multicast*, por ejemplo, se suelen definir como aquellos en los cuales existe un proceso emisor y *m* procesos receptores, donde en algunos casos *m* es determinado o conocido en tiempo de ejecución. Algo similar sucede con las demás operaciones de comunicaciones colectivas.
2. La *semántica* de las operaciones es relativamente variada comparándola con las comunicaciones punto a punto donde siempre hay un proceso que envía y otro que recibe. La operación *scatter*, donde un proceso envía diferentes conjuntos de datos a *m* procesos receptores es esencialmente diferente del *broadcast*, por ejemplo, en lo referente a la cantidad de datos que se comunican entre cada par de procesos (*emisor*, *receptor*) y la cantidad total de datos que se transfieren entre todos los procesos (o la cantidad de datos que se deben transferir por la red de comunicaciones independientemente de la topología u organización que tenga).

Existen múltiples funciones incluidas dentro de las llamadas *comunicaciones colectivas*, algunas de ellas aceptadas en general, pero no existe consenso total respecto de un único conjunto de comunicaciones colectivas. Asumiendo la existencia de *p* procesos que intervienen en una operación de comunicación colectiva (a este conjunto de procesos se lo suele denominar *grupo*), las operaciones más aceptadas como *básicas* son: *Broadcast* (y/o *Multicast*), *Scatter*, *Gather* y *Reduce*. La operación *Barrier* es incluida también como una función entre de las comunicaciones colectivas, pero en realidad no implica la transferencia de ningún tipo de *dato* sino que en realidad establece un

punto de sincronización de todos los procesos que intervienen. En general, desde el punto de vista de la implementación y/o el rendimiento, se podrían distinguir las operaciones que se basan en el broadcast de un único conjunto de datos de las operaciones que se pueden definir en función de múltiples comunicaciones punto a punto.

Entre las operaciones mencionadas previamente, tanto el *Broadcast* (y/o *Multicast*) como el *Barrier* se pueden resolver con un único mensaje de broadcast físico que llegue a todos los procesos que intervienen. En el caso del *Barrier* se podría enviar un "mensaje vacío" o con longitud mínima y el broadcast en este caso debería ser definido como bloqueante, es decir que ningún proceso continúa su ejecución posterior a la operación de comunicación sino hasta que todos han concluido de *enviar* (un proceso) y *recibir* (los p-1 procesos restantes) en términos del mensaje broadcast con el cual se implementa el *Barrier*. La mayoría (sino *todas*) las demás operaciones se pueden definir en función de múltiples comunicaciones punto a punto y en general este tipo de implementaciones tiende a mejorar sensiblemente el rendimiento en las redes de interconexión estáticas con organizaciones bidimensionales (tales como las *mallas* o *toros*) o en las redes con cableado basado en *switches* que son bastante utilizadas en las instalaciones Beowulf. Entre las operaciones mencionadas previamente, la implementación de *Scatter*, *Gather* y *Reduce* utilizando múltiples comunicaciones punto a punto es inmediata. En realidad, la implementación de todas las operaciones colectivas es *inmediata* utilizando comunicaciones punto a punto, pero desde el punto de vista del rendimiento las diferencias pueden ser dadas en órdenes de magnitud, dependiendo del hardware y de la longitud de los mensajes involucrados.

La estimación de rendimiento para las operaciones implementadas con múltiples operaciones punto a punto se deriva de la estimación misma de las operaciones punto a punto involucradas, adaptando el modelo de rendimiento de la operación colectiva que se trate de acuerdo con la implementación. Si, por ejemplo, la operación *Scatter* de n datos entre p procesos se implementa con p-1 operaciones punto a punto, el tiempo total estimado será,

$$t_{Sc}(n, p) = (p\text{-}1)\, t(n/p) \tag{2}$$

y aplicando la Ec. (1),

$$t_{Sc}(n, p) = (p\text{-}1)\, (\alpha + (n/p)/\beta) \tag{3}$$

Si la implementación implica la construcción de un *árbol abarcativo* (*spanning tree*) de los p procesos involucrados [22], la modelización del rendimiento involucra el logaritmo de la cantidad de procesos, donde la base del logaritmo depende del tipo de árboles que se construyen. En el caso de la construcción de un árbol binario que abarca a los p procesos, de los cuales p-1 procesos deben recibir información por el *Scatter*,

$$t_{Sc}(n, p) = 2 \log_2(p)\alpha + (2(n/p)(p\text{-}1))/\beta \tag{4}$$

En todos los casos es muy difícil conocer *a priori* cómo es implementación de cada una de las operaciones y esta dificultad se traduce en la imposibilidad del usuario de las bibliotecas de pasaje de mensajes para estimar el rendimiento de sus aplicaciones de manera efectiva. En el caso particular de las operaciones que necesariamente involucran broadcast o multicast de datos (como las mismas operaciones *Broadcast* y *Multicast*), ya no sería posible conocer *a priori* si la implementación efectivamente realiza y/o aprovecha la posibilidad de hacer broadcast físico de los datos en la red de interconexión. Aunque en principio parezca una característica poco importante, la forma en que se implementa la operación *Broadcast* se torna bastante importante en el contexto de

los clusters que se utilizan para cómputo paralelo dado que, desde el punto de vista de la algorítmica paralela:

- Muchas de las aplicaciones paralelas numéricas y específicamente las aplicaciones del álgebra lineal para las que se utilizan los clusters actualmente involucran esta operación y por lo tanto su rendimiento se verá directamente afectado por el rendimiento de la operación *Broadcast*.
- Muchas de las aplicaciones que se paralelizan podrían simplificarse si se expresan en términos de la operación *Broadcast*.
- Aunque no está probado en general, las operaciones *Broadcast* y *Multicast* tienen mayor probabilidad de aparición y frecuencia de utilización en las aplicaciones paralelas que el resto de las comunicaciones colectivas.

Además, desde un punto de vista mucho más cercano al hardware y al análisis de rendimiento del hardware mismo, la operación *Broadcast* tiene al menos tres ventajas con respecto a las demás operaciones:

1. Es posible su implementación directa en hardware dado que los clusters utilizan de manera casi uniforme las redes de interconexión Ethernet por su bajo costo, y aparentemente esto se va a seguir manteniendo.
2. No depende del cableado, en particular de la utilización de switches que permite tener múltiples comunicaciones punto a punto simultáneas. El broadcast de Ethernet está definido por la norma en el nivel más bajo de las comunicaciones, todas las formas de cableado (incluyendo a los switches) deben, por lo tanto, tener implementado el broadcast de los datos.
3. Es tan escalable como la definición misma de la norma Ethernet, que permite la interconexión de una cantidad realmente muy grande de máquinas al menos desde el punto de vista de la mayoría de las actuales instalaciones Beowulf que se mantienen debajo de las 100 computadoras interconectadas por switches de muy alto costo y rendimiento (comparado con el costo de una placa de interfase Ethernet).

Por las razones enumeradas, el estudio del rendimiento de la herramienta propuesta se “reduce” a la operación *Broadcast*. Como para la mayoría de las demás operaciones, existen varios modelos de rendimiento y *a priori* es muy difícil aplicar uno de ellos sin conocer los detalles de la implementación de esta operación de comunicaciones. En todos los casos, siempre se incluyen los índices anteriores de latencia y ancho de banda asintótico, pero como en el caso de la operación *Scatter* que se menciona antes, aparecen en los modelos otras características que afectan el rendimiento. Como mínimo, suele estar representado en el modelo de rendimiento la cantidad de procesos que intervienen en la operación de *Broadcast*, muchas veces de manera directa multiplicando a la latencia y/o relacionado con el ancho de banda asintótico. La forma en que se pretende “captar” el comportamiento del rendimiento de la operación Broadcast es la siguiente:

1. Primero, para sincronizar los tiempos de espera hasta que todos llegan a la operación (Broadcast o Multicast) se hace un Barrier entre todos los procesos participantes.
2. Luego se lleva a cabo el Broadcast, un proceso designado como “root” que ejecutará desde la máquina en la que se inician las pruebas será el encargado de hacer el envío de los datos y todos los demás deberán recibir. (En el caso de PVM el "root" no recibe y en caso de MPI sí lo hace).
3. Cada proceso receptor medirá el tiempo que tarda en recibir y luego enviará este valor al "root". Este recoge todos los tiempos y calcula cuál fue el mayor de todos, que se toma como el tiempo total de comunicaciones del Broadcast.

La Fig. 3 muestra el pseudocódigo para llevar a cabo la experimentación. El tiempo que busca y muestra la herramienta es en realidad el promedio de muchos mensajes Broadcast individuales que se promedian.

```
P1: /* Root */
  Initialize
  Sync Barrier
```

```
Pi: /* (i!=1) - Receiver */
  Initialize
  Sync Barrier
```

```
S = Get timer                    S = Get timer
Bcast/Mcast                      Bcast/Mcast
E = Get timer                    E = Get timer
Recv all acks                    ACK = E - S
M = MAX(E-S,ACK2,ACK3,...)       Send ACK
```

Figura 3: Pseudocódigo de Medición del Tiempo de un Mensaje Broadcast.

3.- Herramienta de Evaluación Automática de PVM y MPI

Ha sido desarrollada para ejecutar en entorno Unix, y ha sido utilizada sobre diferentes versiones de Linux y sobre Solaris 7. En principio se orienta a los sistemas Unix, pero debido a que existen diferencias entre las implementaciones propietarias del mercado, debe ser probada para asegurar su correcto funcionamiento. Como se mencionó anteriormente, hace uso de PVM (Se ha probado sobre la versión 3.3.11 y 3.4) y de MPI (Se ha probado sobre versión LAM-MPI 6.5.2 y MPICH 1.2.1). La utilización de la herramienta consta de una serie de pasos, que se detallan en las subsecciones que siguen.

3.1 Instalación y configuración

Primero se deben definir algunas variables de entorno y proveer como parámetro de instalación un archivo donde se describen todas las computadoras (nodos o *hosts*) sobre las cuales se ejecutará. Una vez lanzado este paso se generan los ejecutables, *scripts* y archivos necesarios para las pruebas y luego se distribuyen. En el caso de ser una red heterogénea, para realizar este paso se deben agrupar lógicamente los equipos que tengan la misma compatibilidad de binarios y luego ejecutar el paso en cada computadora "representante" del grupo (una computadora cualquiera del grupo). Por ejemplo si se tiene una red con 3 máquinas Linux y 2 Solaris la instalación se deberá hacer desde una de las máquinas con Linux y una de las máquinas con Solaris 2. Desde cada una de esas computadoras se distribuyen los binarios necesarios a las de igual plataforma.

3.2 Lanzamiento de las pruebas

Una vez configurado el entorno, compilados y distribuidos los ejecutables se pueden lanzar las pruebas. Previo a este paso es posible personalizar varios parámetros como son los tipos de ruteos, los modos de empaquetamiento, la cantidad de mediciones, etc. Los experimentos son ejecutados desde una única computadora (cualquiera) en el cual se almacenarán los resultados.

3.3 Presentación de los resultados

Los primeros resultados corresponden a las pruebas de comunicación de bajo nivel (usando el comando ping, protocolo ICMP). Para saber cuál es el rendimiento máximo que se puede alcanzar, se realizan previamente a las pruebas con PVM/MPI la evaluación de latencia y del ancho de banda que se logra usando el comando ping. Para estas pruebas se muestran el mínimo startup y máximo ancho de banda logrado con todos los hosts. Se debe tener en cuenta que no se analiza la pérdida de paquetes debido a que el comando ping usa ICMP que es un protocolo no confiable

Se muestran también los resultados correspondientes a las comunicaciones punto a punto (utilizando PVM y la implementación MPI que se disponga). Se muestran con respecto a los valores de startup de las comunicaciones: a) configuración y el nodo con el cual se logra el mínimo y su

valor, b) configuración y valor del mínimo local, que se obtiene comunicándose con la misma computadora que realiza las pruebas, es decir que no se utiliza la red, c) configuración y nodo con el cual se tiene el máximo y su valor, y d) configuración y valor del máximo local. Se denomina "configuración" a la combinación de tres parámetros: forma de ruteo, modo de empaquetamiento de los datos y cantidad de datos. Con respecto a los valores de ancho de banda se muestran: a) configuración y nodo con el cual se logra el máximo y su valor, b) configuración y valor del máximo ancho de banda local, c) configuración, nodo y máximo para la máxima cantidad de datos que se envían, y d) configuración, y máximo para la máxima cantidad de datos que se envían localmente.

Con respecto a las comunicaciones colectivas se muestran los resultados correspondientes a las operaciones MPI Broadcast, PVM Broadcast y PVM Multicast, y las pruebas se llevan a cabo con todos los nodos disponibles. Con respecto al tiempo de startup, se muestran: a) configuración con la cual se logra el mínimo y su valor, b) configuración con la cual se logra el máximo y su valor. Los valores que se muestran con respecto al ancho de banda son: a) configuración con la cual se logra el máximo y su valor, b) configuración y máximo para la máxima cantidad de datos que se envían.

4.- Resultados de Evaluación Sobre Redes Instaladas

Las características más importantes del entorno de experimentación sobre el cual se llevaron a cabo las pruebas son:

- Se utilizaron dos redes de computadoras, una homogénea y otra heterogénea. La red homogénea es de 100Mb/s con cableado UTP Cat. 5 (100BaseTX) con 1 switch en modo FDX (Full duplex), la red con máquinas heterogéneas es de 10 Mb/s con cableado UTP (10BaseT) con 2 hubs. Los detalles de cada máquina se dan en la Tabla 1 y en la Tabla 2 respectivamente.
- En el caso de PVM se utilizó la 3.3.11. Dado que se ha respetado la compatibilidad y como las operaciones que se utilizan son básicas, se podría utilizar cualquier versión posterior.
- En el caso de MPI, se utilizó LAM-MPI 6.5.2. Tal como es usual, la implementación de MPI que se utilice no es un problema, dado que todas deben implementar la norma MPI, pero las incompatibilidades entre las implementaciones surgen a nivel de las herramientas desarrollo y ejecución de programas con MPI.

O.S.	CPU	CPU Clk	Mem	Red
Linux 2.2.12-20	PC-PIII	700 MHz	64 MB	DEC-Tulip 100BaseTX

Tabla 1: Máquinas de la Red Homogénea (Ocho Computadoras).

O.S.	CPU	CPU Clk	Mem	Red
Linux 2.2.12-20	PC-K6.2	500 MHz	124 MB	NE2000-PCI
Linux 2.4.18	PC-K6.2	500 MHz	126 MB	3Com 3c59x-PC
Linux 2.2.12-20	PC-Celeron	266 MHz	128 MB	NE2000-PCI
Solaris 7 for Intel	Pentium-S	100 MHz	32 MB	NE2000-ISA

Tabla 2: Máquinas de la red Heterogénea.

4.1 Resultados Obtenidos

Inicialmente se mostrarán todos los datos correspondientes a la red de máquinas heterogénas, y luego los de la red homogénea. En la red heterogénea, los datos que muestra la herramienta para las evaluaciones con el comando ping son:

Max. startup = 0.000590 **Min. startup** = 0.000312
MB/s max. = 1.128970 **MB/s max. de max. data** = 1.091019

que se corresponden con el máximo y mínimo tiempo de latencia con el comando ping y el máximo ancho de banda y el ancho de banda obtenido con la máxima cantidad de datos posibles con el comando ping respectivamente. Tanto el tiempo de latencia como el ancho de banda obtenido con el comando ping son relativamente apropiados para una red Ethernet de 10Mb/s.

Los resultados correspondientes a las pruebas punto a punto en la red heterogénea utilizando la biblioteca PVM son:

Max. startup = 0.067233 **Min. startup** = 0.005811 **Local min. startup** = 0.004150
MB/s max. = 0.903852 **MB/s max. de max. data** = 0.903852

donde todos los tiempos de startup de las comunicaciones son muy grandes comparados con los que se tienen con el comando ping en la misma red con las mismas máquinas. La sobrecarga que impone PVM en este aspecto es de varios órdenes de magnitud mayor que la del hardware y de los protocolos de bajo nivel como ICMP. Se debe hacer notar, por ejemplo, que el mínimo tiempo de startup local (es decir sin utilizar la red) ya es de aproximadamente 0.004 segundos, con lo cual se confirma que la sobrecarga se debe a la biblioteca PVM. Aunque no se muestra aquí por razones de espacio, la herramienta sí muestra que el máximo startup se tiene con una computadora y el mínimo con otra, por lo tanto, la gran diferencia entre los valores de startup (máximo y mínimo) depende de las computadoras involucradas en las comunicaciones y a su vez de su capacidad de cómputo relativa. La degradación con respecto al ancho de banda no es tan grande, y dado que los valores de MB/s coinciden, el ancho de banda máximo se logra con la máxima cantidad de datos.

Algunos de los resultados correspondientes a las pruebas punto a punto en la red heterogénea utilizando la biblioteca LAM/MPI son:

Max. startup = 0.001701 **Min. startup** = 0.000459
MB/s max. = 0.968043

donde se puede notar que la sobrecarga de LAM/MPI es menor que la de PVM y se confirma por la diferencia entre los valores de startup que este tiempo depende de la velocidad relativa de las computadoras involucradas. el ancho de banda máximo es un poco mejor que el de PVM pero se confirma la tendencia en cuanto a que si los mensajes son suficientemente grandes no se pierde rendimiento en las comunicaciones.

Con respecto a las comunicaciones colectivas, algunos de los resultados en la red heterogénea utilizando la biblioteca PVM son:

Max. startup = 0.003940 **Min. startup** = 0.002429
MB/s max. = 0.315892

Se debe notar aquí que el ancho de banda es aproximadamente 1/3 del ancho de banda de las operaciones punto a punto, con lo que puede estimarse que la implementación del broadcast se lleva a cabo con múltiples operaciones punto a punto (en esta red se tiene un total de 4 computadoras, con lo que el broadcast implica tres mensajes punto a punto y por lo tanto se tiene así 1/3 del máximo ancho de banda).

En cuanto a la operación de Broadcast, al utilizar la biblioteca LAM/MPI se tienen:

Min. startup = 0.000877 **MB/s max.** = 0.324864

con lo que el startup se mantiene mucho mejor que el de PVM y el ancho de banda más o menos igual (se puede hacer la misma estimación en cuanto a la implementación).

Toda la experimentación en la red homogénea tiende a confirmar el análisis y los comentarios realizados hasta ahora en cuanto al rendimiento de la red utilizando PVM o LAM/MPI. De las

pruebas punto punto se resumen todos lo datos en la Tabla 3, donde se muestran el mínimo tiempo de startup y el ancho de banda máximo que se logra en la red de ocho computadoras idénticas interconectadas por una red Ethernet de 100 Mb/s con un switch.

	PVM	LAM/MPI
Mín. startup	0.003283	0.00014
MB/s max.	6.703611	9.11142

Tabla 3: Resultados Punto a Punto en una Red Homogénea.

En cuanto al mensaje Broadcast, se verificaron experimentalmente dos realidades, una de las cuales ya se ha visto. Por un lado, el ancho de banda al utilizar PVM en la red homogénea es de 1.129769 MB/s, lo cual confirma que la implementación se mantiene con mensajes punto a punto. Por otro lado, sin embargo, al utilizar LAM/MPI el ancho de banda es de 2.305933 MB/s que es aproximadamente el doble de lo que se podría obtener con mensajes punto a punto. En este caso se confirma lo que la documentación de LAM/MPI indica con respecto a la implementación de los mensajes Broadcast, para los cuales se construye un árbol abarcativo (*spanning tree*) y se puede aprovechar de esta manera la capacidad del switch con el cual se pueden realizar varias comunicaciones punto a punto simultáneas.

5.- Conclusiones y Trabajo Futuro

Aunque las redes de computadoras utilizadas para cómputo paralelo (*clusters*) se están aplicando de manera satisfactoria en múltiples áreas, no solamente se tiene el problema de que en general el rendimiento de las comunicaciones es bajo sino que además en muchos casos no se tiene una forma estándar de estimación del rendimiento. En el caso específico de las comunicaciones colectivas el problema es aún mayor, dado que ni siquiera se tiene un modelo uniforme de estimación de rendimiento para todas las operaciones. Esta dificultad se mantiene aún para una única operación de comunicaciones colectivas, dado que depende de la implementación que adopte la biblioteca de pasaje de mensajes que se utiliza. En este artículo se ha presentado una herramienta que permite la estimación de

- Rendimiento de las comunicaciones punto a punto, utilizando el modelo y la metodología estándar y aceptada para tal fin.
- Rendimiento de la operación *Broadcast* como representativa de las operaciones colectivas, o al menos de las operaciones colectivas que pueden ser implementadas aprovechando el broadcast físico de las redes Ethernet.
- Sobrecarga impuesta por las bibliotecas de pasaje de mensajes PVM o las implementaciones de uso libre de MPI. Se puede decir que en el contexto de los clusters, se incluye a la gran mayoría de las aplicaciones paralelas, dado que las implementaciones propietarias no existen en algunos casos (para algunas máquinas) o son muy costosas y por lo tanto de aplicaciones para las cuales existe una gran posibilidad de financiamiento.

Como una consecuencia inmediata del último ítem se tiene que es posible una comparación directa entre el conjunto de librerías que la herramienta es capaz de evaluar de manera automática.

También se ha presentado en este artículo la utilización de la herramienta en dos redes de computadoras en particular, y se ha mostrado cómo, por ejemplo, es posible la identificación de *irregularidades* o situaciones no esperadas en el caso de la utilización de computadoras heterogéneas. Por ejemplo, se ha visto cómo es posible identificar que el tiempo de latencia o

startup de las comunicaciones es dependiente de la velocidad relativa de las máquinas involucradas.

Las extensiones inmediatas a la herramienta presentada se enfocan hacia:

- Otras implementaciones de MPI. En realidad la metodología e incluso la implementación en MPI es independiente de la implementación por la definición misma de MPI. Sin embargo, la forma en que las aplicaciones MPI se ejecutan sí es dependiente de la implementación y por lo tanto se debe adaptar caso por caso. Sin embargo, esta adaptación es mínima considerando que es solamente la forma en que los programas paralelos construidos en base a una implementación MPI se ejecutan en una red de computadoras y todo lo demás es invariante (la metodología y las operaciones de MPI)
- Mayor automatización de la evaluación. Más específicamente en cuanto a la operación *Broadcast*, se podría intentar identificar la relación de startup y ancho de banda para cada cantidad de computadoras involucradas derivando el modelo e implementación en general.
- Derivación del modelo y/o implementación en general de otras y/o todas las comunicaciones colectivas.

Bibliografía

[1] Anderson T., D. Culler, D. Patterson, and the NOW Team, “A Case for Networks of Workstations: NOW”, IEEE Micro, Feb. 1995.

[2] Baker M., R. Buyya, “Cluster Computing at a Glance”, in R. Buyya Ed., High Performance Cluster Computing: Architectures and Systems, Vol. 1, Prentice-Hall, Upper Saddle River, NJ, USA, pp. 3-47, 1999.

[3] Barbosa J., J. Tavares, J. Padilha, “Linear Algebra Algorithms in a Heterogeneous Cluster of Personal Computers”, 9th Heterogenous Computing Workshop HCW'2000, IEEE Computer Society Press, pp. 147-159, 2000.

[4] Beaumont O., V. Boudet, A. Petitet, F. Rastello, Y. Robert, “A Proposal for Heterogeneous Cluster ScaLAPACK (Dense Linear Solvers)”, IEEE Transactions on Computers, vol. 50, No. 10, pp. 1052-1070, Oct. 01.

[5] Becker D. J., T. Sterling, D. Savaresse, J. E. Dorband, U. A. Ranawak, C. W. Packer, “Beowulf: A Parallel Workstation for Scientific Computation”, Proc. of the International Concerence on Parallel Processing, vol. 1, pp. 11-14, Boca Raton, Florida, Aug. 1996.

[6] Blackford L., J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R. Whaley, ScaLAPACK Users' Guide, SIAM, Philadelphia, 1997.

[7] Dietz H., Linux Parallel Processing HOWTO, Jan 98, disponible en http://www.linuxdoc.org/HOWTO/Parallel-Processing-HOWTO.html

[8] Dietz H., W. Cohen, T. Muhammad, T. Mattox, “Compiler Techniques For Fine-Grain Execution on Workstation Clusters Using PAPERS”, 7th Annual Workshop on Languages and Compilers for Parallel Computing, Cornell University, Aug 1994.

[9] Dietz H., R. Hoare, T. Mattox, “A fine-Grain Parallel Architecture Based on Barrier Synchronization”, Proc. International Conference on Parallel Processing, Vol. I, pp. 247-250, August 1996.

[10] Dietz H., T. Muhammad, J. Sponaugle, T. Mattox, PAPERS: Purdue's Adapter for Parallel Execution and Rapid Synchronization, Purdue University School of Electrical Engineering , Technical Report TR-EE-94-11, March 1994.

[11] Dongarra J., A. Geist, R. Manchek, V. Sunderam, Integrated pvm framework supports heterogeneous network computing, Computers in Physics, (7)2, pp. 166-175, April 1993.

[12] Flynn M., “Some Computer Organizations and Their Affectiveness”, IEEE Trans. on Computers, 21 (9), 1972.

[13] Foster I., Designing and Building Parallel Programs, Addison-Wesley, Inc., 1995. *Versión html* disponible en http://www-unix.mcs.anl.gov/dbpp
[14] Hoare C., Communicating Sequential Processes, Englewood Cliffs, Prentice-Hall, 1986.
[15] Hockney R., M. Berry (eds.), "Public International Benchmarks for Parallel Computers", Scientific Programming 3(2), pp. 101-146, 1994.
[16] Hockney R., C. Jesshope, Parallel Computers 2, Adam Hilger, Bristol and Philadelphia, IOP Publishing Ltd., 1988.
[17] Institute of Electrical and Electronics Engineers, Local Area Network - CSMA/CD Access Method and Physical Layer Specifications ANSI/IEEE 802.3 - IEEE Computer Society, 1985.
[18] Kalinov A., A. Lastovetsky, "Heterogenous Distribution of Computations while Solving Linear Algebra Problems on Networks of Heterogeneous Workstations", HPCN Europe 1999, Springer-Verlag LNCS 1593, pp. 191-200, 1999.
[19] MPI Forum, "MPI: a message-passing interface standard", International Journal of Supercomputer Applications, 8 (3/4), pp. 165-416, 1994.
[20] Pacheco P., Parallel Programming with MPI, Morgan Kaufmann, San Francisco, California, 1997.
[21] Radajewski J., D. Eadline, Beowulf Installation and Administration HOWTO, June 1999.
http://www.beowulf-underground.org/doc_project/BIAA-HOWTO/Beowulf-Installation-and-Administration-HOWTO-12.html
[22] Sathish S. Vadhiyar, Graham E. Fagg, Jack Dongarra. Automatically Tuned Collective Communications. Computer Science Department, University of Tennessee, Knoxville.
[23] Wilkinson B., Allen M., Parallel Programming: Techniques and Applications Using Networking Workstations, Prentice-Hall, Inc., 1999.
[24] LAM/MPI (Local Area Computing / Message Passing Interface) Home Page
http://www.mpi.nd.edu/lam
[25] MPICH Home Page http://www-unix.mcs.anl.gov/mpi/mpich/