Cómputo Paralelo Interclusters: Herramientas y Evaluación de Rendimiento

Fernando L. Romero, Antonio Quijano, Armando De Giusti[†], Fernando G. Tinetti[‡]

III-LIDI, Facultad de Informática, UNLP
50 y 115, 1900, La Plata, Argentina

CeTAD, Facultad de Ingeniería, UNLP
48 y 116, 1900, La Plata, Argentina

fromero@lidi.info.unlp.edu.ar, quijano@ing.unlp.edu.ar, degiusti@lidi.info.unlp.edu.ar, fernando@info.unlp.edu.ar

Resumen

Se presenta en este artículo la experiencia desarrollada referente a la utilización de múltiples clusters para cómputo paralelo. Inicialmente, se presentan algunas consideraciones importantes en cuanto la instalación y configuración del *middleware* o software de soporte necesario para realizar cómputo paralelo en este tipo de arquitectura que es relativamente *nueva*. Una de las premisas que se tiene en cuenta en este contexto es la de utilizar software herramientas relativamente estándar y estable. Una vez que se establece la configuración básica para ejecutar programas paralelos interclusters (en más de un cluster), se presentan los resultados obtenidos en lo referente a la caracterización de las transferencias de datos entre los clusters intervinientes. El rendimiento de las comunicaciones interclusters puede ser considerado como una de las características propias de este tipo de plataformas de cómputo paraelo y, por lo tanto, es importante tener una metodología y/o herramienta de caracterización de las mismas. Finalmente, se presentan algunos resultados interesantes de paralelización de dos aplicaciones sencillas a ejecutarse en computadoras de diferentes clusters. Esta paralelización muestra, por un lado, ejemplos específicos *reales* de ejecución programas paralelos intercluster y por otro lado, la aplicación de algunas ideas preexistentes de balance de carga que son aplicables también en este contexto.

Palabras claves: Comunicación de Procesos, Caracterización de Rendimiento, Sistemas Paralelos y Distribuidos, Paralelismo en Clusters e Interclusters.

Abstract

This paper presents the experience developed using multiple clusters for parallel computing. Initially, some relevant considerations are presented in terms installation and configuration of middleware or supporting software necessary to carry out parallel computing over this relatively new type of architecture. One of the premises taken into account within this context is that of using relatively standard and stable software and tools. Once the basic configuration is established to run interclusters (on more than one cluster) parallel programs, the obtained results regarding the characterization of data transferences among clusters are presented. Interclusters communication performance can be considered as one of the typical characteristics of this type of parallel computing platforms and, thus, it is important to count with a characterization methodology and/or tool of them. Finally, some interesting results of parallelizing two simple applications run in machines of different clusters are presented. On the one hand, this parallelization shows actual and specific results of running interclusters parallel programs and, on the other, the application of some pre-existing ideas of load balance, which are also applicable within this context.

Keywords: Process Communication, Performance Characterization, Parallel and Distributed Systems, Parallelism in Clusters and Interclusters.

[†] Investigador Principal CONICET. Profesor Titular D.E. Facultad de Informática UNLP
[‡] Investigador Asistente CICPBA

1 INTRODUCCION

La utilización de clusters para cómputo paralelo está bien establecida desde hace varios años [3] y actualmente existe una gran cantidad de clusters instalados que están siendo utilizados con software paralelo en producción. El modelo de programación de pasaje de mensajes también ha madurado en definiciones tales como las de PVM (Parallel Virtual Machine) [5] y MPI (Message Passing Interface) [14]. La biblioteca MPI se convirtió rápidamente en uno de los estándares más importantes para el desarrollo y la ejecución de programas paralelos. Esta biblioteca rápidamente se ha implementado en los sistemas de cómputo paralelo más distribuidos o desacoplados como lo son los clusters. De hecho, desde hace varios años existen al menos dos implementaciones de uso libre: MPICH [11] y LAM/MPI [4].

Aunque se puede asociar pasaje de mensajes y PVM y MPI en particular con hardware de cómputo distribuido o MIMD (Multiple Instruction Stream, Multiple Data Stream), PVM y MPI no *imponen* o *presuponen* ninguna característica sobre/de el hardware. Esto permite una gran independencia y portabilidad de los programas paralelos, incluyendo variantes como las de clusters de SMP (Symmetric MultiProcessing) y computadoras paralelas de memoria compartida [15]. En general, las bibliotecas de pasaje de mensajes resuelven básicamente dos problemas técnicos de manera satisfactoria para ser utilizadas:

1. Identificación única de procesos. Los programas paralelos no son mucho más que un conjunto de procesos que se pueden identificar de manera unívoca, independientemente de que se ejecuten en un ambiente de memoria compartida o memoria distribuida, por ejemplo.
2. Transferencia de datos entre los procesos. Si bien se reconoce que las primitivas básica de comunicación son del tipo *send*() y *receive*() punto a punto entre dos procesos, también se suelen incluir otras variantes como las comunicaciones colectivas del tipo de *broadcast*(), por ejemplo, o variantes semánticas de las comunicaciones punto a punto (haciendo alusión explícita a *buffers* de memoria, por ejemplo).

De hecho, en cualquier plataforma de cómputo donde se pueden resolver estos problemas, se podría utilizar una implementación de MPI, por ejemplo. Evidentemente los clusters han sido apropiados desde la perspectiva de las implementaciones de bibliotecas de pasaje de mensajes para cómputo paralelo, y estos dos problemas se resuelven de manera relativamente sencilla. Todo lo relacionado con transferencias de comunicaciones se ha resuelto utilizando protocolos estándares como TCP/IP y no es necesario un gran esfuerzo para asignar y mantener una identificación de procesos para bibliotecas como la más utilizada actualmente de MPI 1.1, donde no hay creación dinámica de procesos. De hecho, PVM también incluye la creación dinámica de procesos y tampoco en este caso es un gran problema mantener actualizada la asociación proceso-identificador en un ambiente distribuido. Evidentemente no es un problema en un ambiente centralizado (como la de un SMP), donde hay un único sistema operativo y, de hecho, cualquier sistema operativo tiene identificación única de procesos además de muchos otros datos de información de estado del sistema.

La propuesta en este artículo es la de plantear la solución al problema planteado por la ejecución de un programa paralelo en más de un cluster o, lo que es igual: la utilización de más de un cluster para ejecutar un programa paralelo, que se considerará sinónimo de "cómputo paralelo interclusters". De una manera o de otra, hay varias alternativas de solución a este problema, algunas de ellas disponibles desde hace bastante tiempo. De hecho, se podría resolver inicialmente con las mismas bibliotecas PVM o implementaciones de MPI si no fuera por los controles de seguridad que se imponen actualmente al tráfico sobre Internet de la mayoría (si no todas) de las instituciones que tienen clusters disponibles para cómputo paralelo. De hecho, la propuesta avanza un poco más en el sentido de no solamente utilizar más de un cluster para un programa paralelo sino de hacerlo con el mínimo costo de instalación y mantenimiento de software y, además, con lo que se supone la menor

sobrecarga (*overhead*) de cómputo y comunicaciones posible. Justamente desde esta perspectiva se analizarán las propuestas existentes (al menos las más importantes o consideradas suficientemente representativas) en la siguiente sección.

Una vez resuelto el problema básico de cómputo paralelo interclusters (con la utilización de más de un cluster) comienza, en realidad, un problema mayor desde la perspectiva de paralelización de aplicaciones y la optimización de rendimiento. En un ambiente de cómputo paralelo interclusters no se puede asumir que todas las comunicaciones punto a punto tienen el mismo rendimiento, por ejemplo. Desde la perspectiva de un proceso que envía, algunos receptores son locales (en la misma red local, el mismo cluster) y otros no, están en otro cluster, al que se llega, en el caso más *usual*, por ruteo IP (normalmente). En cualquier caso, el rendimiento de las comunicaciones (el tiempo necesario para efectuar una transferencia de datos) no será el mismo. Otra de las características inherentes del cómputo paralelo interclusters es la de heterogeneidad de las computadoras a utilizar. Si bien en un cluster normalmente todas las computadoras son iguales (y, en particular, con la misma potencia de cálculo), es muy poco probable que las computadoras de dos o más clusters sean exactamente iguales entre sí. Y esto afecta directamente el balance de carga para que el cómputo sea *equilibrado* en cuanto al tiempo necesario en cada una de las computadoras que se utilizan. Los problemas planteados en cuanto a rendimiento de comunicaciones y balance de carga computacional (y tiempo de cómputo asociado) serán analizados en otra sección de este mismo artículo.

2 TRABAJOS PREVIOS RELACIONADOS

Aunque la terminología varía bastante, dado que no ha sido estandarizada, entre los primeros esfuerzos de utilizar computadoras en más de un cluster se puede encontrar la idea de "clusters geográficamente distribuidos". Entre los primeros esfuerzos se pueden encontrar herramientas o bibliotecas tales como MagPIe del proyecto Albatros [24]. La biblioteca MagPIe fue orientada directamente a la optimización de las funciones de comunicaciones colectivas de MPI en redes de área extensa o extendida (WAN: Wide Area Network) [13] [12]. En algunos otros casos, se pueden encontrar los esfuerzos por hacer diferentes versiones de MPI interoperables, asumiendo que se tienen varios clusters, cada uno con su implementación *propia* (o propietaria o específicamente orientada a un tipo de máquina paralela o cluster). Entre estas iniciativas se pueden mencionar MPI-Connect [7] con su precedente PVMPI [6] y también IMPI [10], que se propuso con mayor generalidad para la interoperabilidad, con la definición de un protocolo apropiado.

La gran mayoría de las propuestas son de finales de la década de 1990, mayormente entre los años 1995 y 2000. Quizás por esta razón, de alguna manera o de otra estas propuestas han sido *absorbidas* o incluidas es lo que hoy se conoce como en Grid Computing [9]. Por ejemplo, el proyecto Albatross se asocia directamente con DAS [24] y el sitio web de DAS indica que ha sido sucedido por DAS-2 y DAS-3 [DAS-3], donde este último (DAS-3) se define como la infraestructura de Grid Computing en Holanda. Sin embargo, parece conveniente en ciertas aplicaciones mantener la propuesta de cómputo paralelo interclusters separado de Grid Computing por al menos dos razones:

- Grid Computing está propuesto como una solución *integral* o *completa* para compartir recursos de cómputo y almacenamiento a gran escala de distribución y de capacidad. Obviamente esto incluye la utilización de varios clusters, pero desde la perspectiva de compartir de manera controlada múltiples recursos disponibles en múltiples instituciones y/o instalaciones de cómputo. Esto incluye también la idea de mantener un gran sistema distribuido más que un gran sistema paralelo (aunque el sistema distribuido pueda utilizarse para cómputo paralelo).
- Grid Computing tiene mucho más que lo necesario para cómputo paralelo interclusters. Por

ejemplo, se intenta proveer SSI (Single System Image) no solamente a nivel de *Single Sign On* (o único punto de identificación y conexión) sino a nivel de proveer y obtener recursos para o de todo el sistema de Grid Computing. Esto implica, por ejemplo, un sistema muy elaborado y complejo para proveer adaptación a los diferentes sistemas de seguridad que se utilicen localmente en cada instalación/institución conectada a grid.

La propuesta de este artículo consiste en mantener la posibilidad de utilizar computadoras de más de un cluster sin tener que instalar toda una infraestructura (o *middleware*) como la de Globus [8]. Un ejemplo sencillo puede ser la colaboración puntual de dos o más instituciones para resolver un problema específico, donde esto no implique la definición e instalación de toda una infraestructura de software específica para el desarrollo de la solución, además de lo que realmente es necesario: desarrollar la solución. Cada una de las instituciones (o, más específicamente, cada uno de los clusters a utilizar) no debería ser mayormente afectado en cuanto a infraestructura de software para la colaboración en cuanto a cómputo paralelo. En cierta forma, puede considerarse que esto *restringe* o *limita* la escalabilidad de aplicaciones, pero sin lugar a dudas también reduce la complejidad de instalación y mantenimiento de la infraestructura de software necesaria. En el otro extremo, se podría mencionar la colaboración explícitamente *ad hoc* como en [1], en el sentido de desarrollar no solamente la aplicación sino también las comunicaciones y el control de la aplicación paralela. Aunque sin lugar a dudas esta es la forma con menor requerimiento de infraestructura de software *a priori*, también involucra un alto costo de desarrollo *extra* sobre el programador. Las comunicaciones, por ejemplo, se deberían llevar a cabo usando métodos que son relativamente rudimentarios para cómputo paralelo como el desarrollo directo sobre la biblioteca de *sockets BSD*. En este contexto, el objetivo es mantener un mínimo estable de desarrollo y ejecución de programas paralelos como el de MPI sin tener que recurrir a sus implementaciones para Grid Computing, por ejemplo, donde se requiere, además, todo el soporte que corresponde, como el que provee específicamente Globus.

Desde hace algún tiempo, se han estudiado características específicas de cómputo paralelo interclusters tales como la del problema generado por las interconexiones no dedicadas y los problemas de seguridad involucrados [2] [18]. A modo de resumen, en estos trabajos previos se reportan algunos detalles técnicos importantes a tener en cuenta para cómputo paralelo interclusters:

- En los ambientes no dedicados, muchos de los problemas de disponibilidad de las computadoras de los clusters a utilizar son propios de la falta de control sobre los mismos, no de las comunicaciones o estabilidad de las computadoras.
- Aunque la interconexión de los clusters no es dedicada, siempre hay conectividad entre los clusters a utilizar, salvo algunas excepciones relativamente muy poco frecuentes. Esto se debe básicamente a que la interconexión está involucrada con el tráfico de Internet, que es mantenido y monitoreado independientemente de la utilización de cómputo paralelo interclusters.
- El rendimiento que se podría considerar como *raw* (medido a partir de tráfico ICMP, por ejemplo) de las comunicaciones entre los clusters no es constante dado que no es dedicado, pero en general es muy cercano al máximo absoluto, al menos para transferencias de relativamente pocos datos (decenas de K Bytes, por ejemplo). Como es de esperar, el rendimiento para las comunicaciones interclusters fluctúa dependiendo de días y horarios.
- Los mecanismos básicos de seguridad que se imponen en las instituciones (y que son de uso común en casi todas las instalaciones de computadoras) normalmente impiden la utilización directa de implementaciones de MPI como MPICH y LAM/MPI. Estas implementaciones imponen un patrón de tráfico TCP/IP que normalmente es cancelado por los *firewalls* y/o mecanismos de seguridad de las instituciones.

3 UN SOPORTE SIMPLE PARA COMPUTO PARALELO INTERCLUSTERS

Quizás el soporte más simple para el cómputo paralelo interclusters es el que proveen las propias implementaciones de MPI que, *a priori*, permiten la utilización de múltiples computadoras independientemente de su ubicación geográfica y/o en clusters. Esta posibilidad debe ser descartada de plano por los múltiples problemas de seguridad y administración que involucra. Desde la perspectiva de seguridad, se deberían *relajar* los niveles de control, al menos desde la perspectiva de los *firewalls* que son de uso extendido en todas las instituciones. Como se reporta en [17], tanto el o los protocolos (TCP y/o UDP) como la cantidad y el *tipo* o *clase* (privilegiados o no) de *puertos* a utilizar por las aplicaciones que utilizan implementaciones de MPI no es configurable. Esto llevaría a dejar sin protección o sin control el tráfico entre las computadoras involucradas de los diferentes clusters. Aún en el caso de considerar que eliminar el control de tráfico entre las máquinas no es suficientemente peligroso, sí es un problema de administración. Al menos un administrador por institución debe encargarse de quitar estos controles para que las aplicaciones MPI puedan ser ejecutadas. Este problema se suele complicar debido a que normalmente los administradores de la seguridad no tienen relación directa con cómputo paralelo y que en las instituciones con control más distribuido se involucra a más de un administrador. Por lo tanto, mantener el uso directo de MPI no es sustentable desde el punto de vista técnico por la seguridad ni operativo por la necesidad de recurrir a múltiples administradores de las múltiples instituciones involucradas.

La *siguiente* alternativa para ejecutar aplicaciones paralelas interclusters basadas en MPI podría ser la utilización de algunas de las herramientas enumeradas en la sección anterior: MagPIE, MPI-Connect, PVMPI, o IMPI. Algunas de las alternativas podrían considerarse con el *costo* agregado de instalación de la o las bibliotecas más la recompilación de los programas paralelos. Sin embargo, como se comentó en la sección anterior, el problema más significativo es que la mayoría (sino todas) de estas herramientas simplemente se *trasladaron* al ambiente de Grid Computing. Esto significa que, como mínimo no tienen soporte ni actualizaciones para los clusters (o interclusters) actuales. De hecho, en el contexto de Grid Computing directamente existen implementaciones de MPI que evitarían el uso de otra biblioteca. Como también se comenta en la sección anterior, el objetivo es evitar la instalación de la infraestructura de una implementación para Grid Computing por el propio costo de la instalación y además para evitar el costo en tiempo de ejecución para el acceso a recursos (CPU, memoria, etc.) vía Grid Computing.

Tal como se adelanta en [18] a nivel preliminar, la idea es recurrir a la utilización de una red privada virtual, o VPN (*Virtual Private Network*) como *middleware* asociado en cierta forma a la implementación de MPI que se utilice. Desde un punto de vista técnico, todo lo que *necesita* MPI para ser utilizado en un cluster es que se tenga conectividad TCP/IP como en una red local. Esto es, justamente, lo que provee una VPN en cualquiera de sus versiones o implementaciones [25] [23]. Sin lugar a dudas, se tienen costos por la utilización de una implementación de VPN para cómputo paralelo interclusters:

- Se debe instalar y configurar el software de la implementación de VPN que se haya elegido. Normalmente no es muy complejo ni requiere muchos conocimientos previos. Casi todas las distribuciones de Linux tienen o incluyen alguna implementación. Este es el caso de OpenVPN, que es el que se utilizó para el trabajo de este artículo y cuya tarea de instalación y configuración se reporta en [19].
- Desde la perspectiva de seguridad o en realidad lo que se puede ver afectado por las controles de seguridad existentes vía *firewalls*, las implementaciones de VPN no impone mayores complicaciones. Normalmente se tiene un esquema cliente/servidor con un puerto bien conocido para el servidor, al cual los clientes hacen requerimientos. Esto implica que todo el

tráfico se "concentra" (de allí en parte el nombre de "túnel" o "entubamiento" o *tunnel* que suele aparecer en la bibliografía de VPN) en un puerto bien conocido del servidor que se instale. Esto de hecho simplifica lo relacionado con la seguridad, dado que la configuración de los *firewalls* se orienta, justamente, al control sobre números de IP y puertos a ser utilizados en tráfico TCP/UDP sobre IP. Al establecer un único IP (el del servidor) y un único puerto (el puerto bien conocido del servicio de VPN, que de hecho es configurable) las tareas son mínimas. De hecho, los controles actuales de tráfico *peer to peer* se pueden evitar con cierta sencillez dado que el servidor de VPN se puede configurar para que utilice puertos privilegiados en vez de los no privilegiados que normalmente se utilizan y se filtran en los *firewalls* por el tráfico *peer to peer*.

- Evidentemente en tiempo de ejecución existe una sobrecarga de procesamiento y en cierto modo también de tráfico de datos por el entubamiento de las comunicaciones sobre el esquema de cliente/servidor sobre un puerto bien conocido. En principio, en este artículo se pondrá énfasis en la factibilidad, por lo tanto no se estudiará específicamente el problema de la sobrecarga. Sin embargo, se puede estimar a priori que la utilización de una VPN implica menor sobrecarga que una infraestructura como la de Grid Computing.

La Fig. 1 muestra esquemáticamente el ambiente de ejecución de programas con MPI a partir de la utilización de una VPN. Desde la perspectiva del programador y sin tener en cuenta lo relacionado con el rendimiento de las comunicaciones, no hay ningún cambio, ni siquiera es necesario recompilar, dado que los binarios generados son independientes de que se utilice VPN o se ejecute en un cluster de computadoras en una LAN. En la Fig. 1 no se muestran las conexiones *reales* (las que resuelve la implementación de VPN instalada) entre las máquinas sino que con líneas de punto se muestran las conexiones vía VPN, como se utilizan desde los programas con MPI. El tráfico entre las computadoras de diferentes clusters se resuelve normalmente vía el que se define como *servidor* de VPN, dependiendo de la implementación y la configuración utilizadas.

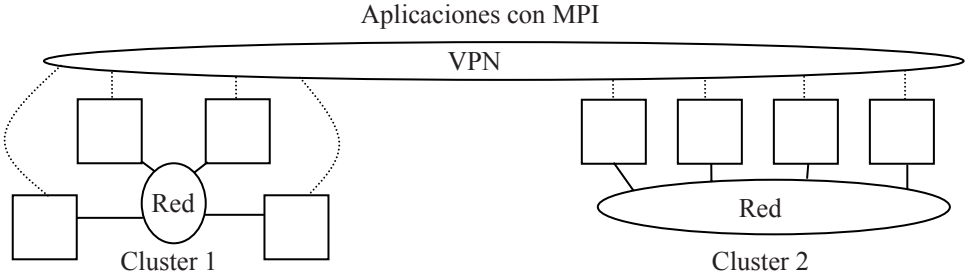

Figura 1: Infraestructura Sencilla para Cómputo Paralelo Interclusters con VPN.

4 EXPERIMENTACION: EVALUACION DE COMUNICACIONES

Tal como se reporta en [18], se utiliza el método sencillo de ping-pong de mensajes para evaluar el rendimiento de las comunicaciones entre dos procesos (en este caso en dos clusters diferentes), siguiendo el modelo de tiempo para las transferencias de datos

$$t(n) = \alpha + \beta\, n$$

donde n es la cantidad de datos, α es el tiempo de latencia (mínimo tiempo siempre se necesita para las comunicaciones) y β es la inversa del ancho de banda, o costo en tiempo por dato a comunicar. Uno de los primeros problemas encontrados, fue que no es posible utilizar cualquiera de las implementaciones de MPI sobre la VPN. En particular, LAM/MPI tiene problemas de ejecución, probablemente por las transferencias de datos entre los procesos lamd que ejecuta en cada una de las máquinas utilizadas. Para evitar el estudio de las razones de este problema, se recurrió a una implementación de MPI totalmente *estática* en el sentido de que no hay procesos de *administración*

o *intermedios* (del tipo de los lamd para LAM/MPI) para la ejecución de aplicaciones, MPICH, versión 1.2.4. Si bien en cierta forma se contradice la idea expresada en cuanto a que cualquier implementación de MPI se podría usar sobre la VPN, siempre es posible:

1. Encontrar el problema por el cual no funciona una implementación en particular y resolverlo (o hacer el requerimiento a quienes producen la implementación).
2. Encontrar una implementación de MPI que funcione sobre la VPN. En este caso, al menos para se recurre a MPICH que es de las implementaciones de MPI más utilizadas y respetadas de entre las de uso libre.

El entorno de experimentación elegido fue el más sencillo posible en cuanto a cantidades de computadoras: dos PCs con Linux, cada una en una red local diferente. Las dos redes locales involucradas son en realidad dos subredes de la 163.10.xx.yy de la UNLP, una en el ámbito de la Facultad de Informática y la otra en el ámbito de la Facultad de Ingeniería. La red de interconexión entre las dos redes locales no es exclusiva y, por lo tanto, se comparte o compite por el ancho de banda disponible entre estas redes con el tráfico usual de otras aplicaciones relacionado con Internet. La cantidad de computadoras con el que se comparte esta interconexión está en el orden de las centenas. La cantidad y el tráfico que involucran los experimentos ping-pong fueron determinados de forma tal que:

- No utilicen más del 5% de 10Mb/s que se asume como el máximo ancho de banda disponible entre las redes locales, con múltiples *routers* intermedios, algunos de los cuales utilizando placas Ethernet de 10 Mb/s. En cualquier caso, este tráfico es muy *conservador* en el sentido de evitar al máximo cualquier congestión de tráfico en la red de interconexión compartida.
- Los experimentos se distribuyen de manera uniforme durante todo el tiempo de ejecución, de forma tal que se monitoricen tiempos de uso normal de las redes locales y de tráfico de Internet con el que se compite con otras aplicaciones en el tramo de interconexión entre los clusters.
- El 50% de los experimentos se *orienta* a identificar la latencia de los mensajes entre dos procesos de una aplicación paralela y el otro 50% se *orienta* a la identificación del ancho de banda disponible o *posible* entre los procesos. La longitud de los mensajes de los experimentos orientados a latencia se establece en 8 bytes y la longitud de los mensajes orientados a ancho de banda se establece en 20000 bytes (básicamente para no provocar ráfagas de uso muy intensivo sobre la red compartida, como se explica en el primer punto).
- Los experimentos se llevaron a cabo durante aproximadamente 10 días corridos, para tener datos de días y horas de uso normal de las computadoras y las redes intermedias y también de días y horas con relativamente poco de uso de las computadoras y redes involucradas.

La Fig. 2 muestra el histograma de la distribución de los tiempos de latencia de los experimentos, donde se puede observar que la gran mayoría de los mensajes tiene una latencia de entre 1 y 5 milisegundos. Es claro que esta latencia es muy elevada para las capacidades de cómputo de las PCs actuales (que son del orden de Gflop/s), pero en cierto modo es muy importante identificar con experimentos tan sencillos que alrededor del 95% de las comunicaciones tienen valores de latencia en este rango. Esta información es particularmente útil para las aplicaciones paralelas, dado que dan una idea de la granularidad mínima: no tiene sentido comunicarse entre tiempos de cómputo menores a los de la latencia. La Fig. 3 muestra la distribución del ancho de banda de los experimentos realizados con 20000 bytes, donde como en el caso de la latencia se puede observar que la mayoría está concentrado en un rango de valores relativamente pequeño. A modo de resumen de los valores mostrados en la Fig. 3, la gran mayoría (alrededor de 95%) de los experimentos se lleva a cabo con entre 50 y 58 KB/s. Es de destacar que durante todo el tiempo de ejecución de los experimentos no hubo que reinstalar ni reconfigurar OpenVPN, la conectividad entre las computadoras se mantuvo dentro de la VPN, a diferencia de lo que había sucedido en los experimentos reportados en [20].

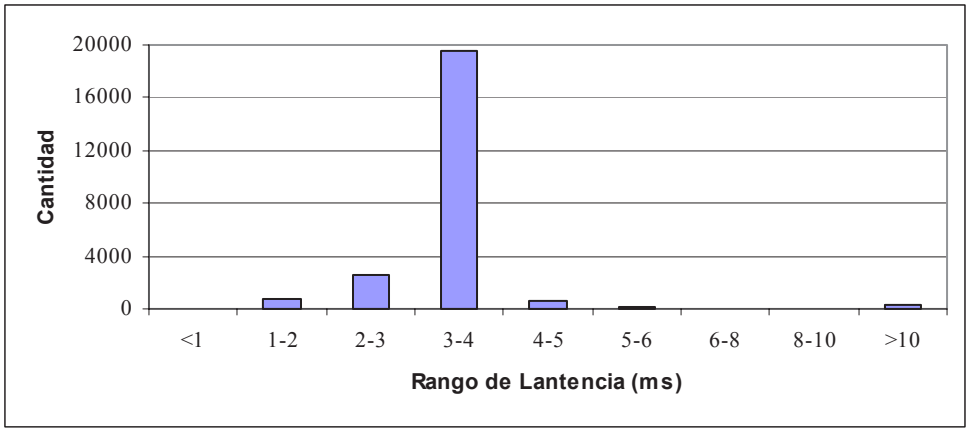

Figura 2: Distribución de Tiempos de Latencia (*Startup Time*) con MPI en VPN.

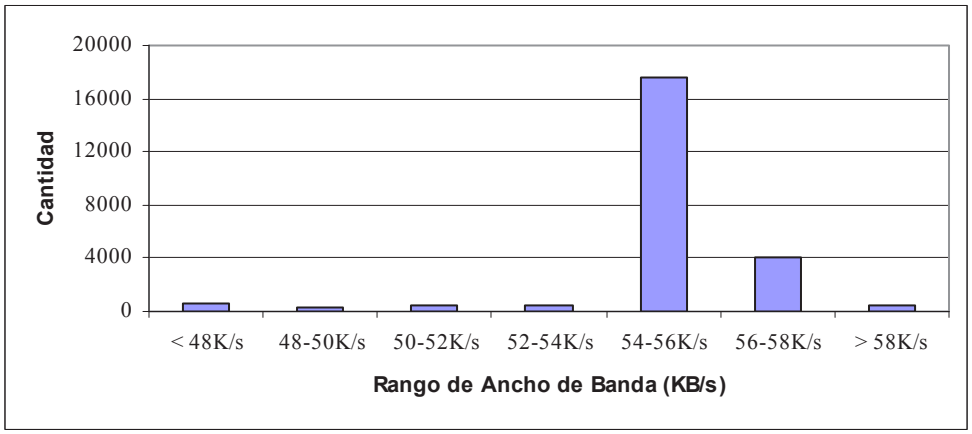

Figura 3: Distribución de Anchos de Banda con MPI en VPN.

5 EXPERIMENTACION: PARALELIZACION DE APLICACIONES

Como ya se ha comentado, el hecho de que sea factible el cómputo paralelo interclusters no resuelve los problemas asociados a la paralelización de aplicaciones. De hecho se podría afirmar que por un lado aporta un horizonte nuevo de paralelización y sus problemas asociados. Uno de ellos es la asimetría o diferencia en las comunicaciones *intra* e *inter* clusters que se ha comentado. A partir de los resultados que se muestran en la sección anterior se tienen datos más precisos al menos del entorno sobre el que se llevó a cabo la experimentación:

- El rendimiento de las comunicaciones interclusters no es constante, aunque los rangos son reducidos.
- El rendimiento de las comunicaciones interclusters es varias veces menor que el de las comunicaciones dentro de un mismo cluster. Solamente a modo de ejemplo, en un cluster interconectado con Ethernet de 100 Mb/s se tienen aproximadamente 10 MB/s entre procesos, lo cual es muy superior al rango de 50-58 KB/s que se muestra en la Fig. 3.

Las aplicaciones denominadas *altamente paralelas* o *embarazosamente paralelas* (*embarrassingly parallel*) son las primeras opciones para ser resueltas en cualquier plataforma de cómputo paralelo. Esto se debe a que en realidad paralelizar estas aplicaciones no requiere casi ningún esfuerzo más allá de la propia codificación del programa, por ejemplo con MPI. Por otro lado, estas aplicaciones normalmente requieren muy pocas comunicaciones durante el tiempo de cómputo o, puesto de otra forma, se pueden paralelizar con granularidad muy gruesa. Esto las hace *mejores* en cuanto a la obtención de rendimiento óptimo o cercano al óptimo en ambientes con muy bajo rendimiento de

comunicaciones, como el que se utiliza para la experimentación en este artículo.

Se han elegido dos aplicaciones muy conocidas pero con diferentes características de paralelización con la finalidad de evaluar rendimiento: integración numérica (cálculo del número π) y cálculos asociados con el conjunto de Mandelbrot. En ambos casos, los problemas de cálculo asociados son muy sencillos de paralelizar dado que se pueden identificar *partes* de cómputo independiente de manera natural, sin recurrir a cambios en el tipo de cómputo secuencial básico. Nuevamente el ambiente de experimentación es el de mínima complejidad en cuanto a cantidad de computadoras: una computadora en cada uno de los clusters. Dado que es importante la capacidad de cómputo para estos experimentos, en la Tabla 1 se detallan las características técnicas más importantes de las dos computadoras utilizadas, donde se puede notar que son muy diferentes en cuanto a capacidad de procesamiento y almacenamiento.

Tipo	CPU	Memoria RAM	Sistema Operativo
PC	Pentium 4 2.4 GHz	1 GB	Linux 2.4.18-14
PC	Pentium II 400 MHz	128 MB	Linux 2.4.18-14

Tabla 1: Características de las Computadoras Utilizadas en los Experimentos.

Un problema común en el contexto de cómputo paralelo interclusters es el de la heterogeneidad en cuanto a la capacidad de cómputo de las computadoras que se utilizan. Desde la perspectiva de la paralelización, esto genera un problema de balance de carga implícito en la suposición de que a todos los procesadores se les asignará la misma cantidad de operaciones a realizar. Se debe notar que esto no es un problema de paralelización sino de rendimiento (quizás implícito) de la paralelización. Para balancear la carga o, más específicamente, para asignar la carga de procesamiento acorde a la capacidad relativa de cada procesador se tienen que resolver dos problemas:

1. Conocer las diferencias relativas de capacidades de cómputo, es decir para cada par de computadoras cuánto mejor o peor es una respecto de la otra.
2. Asignar la cantidad de cómputo de cada computadora de acuerdo con su capacidad relativa respecto de las demás.

El primer problema es sencillo de resolver utilizando la idea ya planteada en [21] y [16], es decir ejecutando el mismo problema pero con tamaño reducido en cada una de las computadoras y relacionando directamente los tiempos de cómputo. Con estos cálculos no solamente se puede definir el balance de carga sino también la evaluación de rendimiento. El segundo de los problemas planteado normalmente depende de la aplicación, es decir que se resuelve caso por caso.

5.1 Cálculo de π

Una forma numérica de calcular el número **π** se lleva a cabo vía integración numérica con el cálculo

$$\pi = h \times \sum_{i=0}^{n-1} 4/(1+x_i^2) \qquad (1)$$

donde $h = 1/n$, n es la cantidad de puntos a utilizar en la integración numérica y $x_i = h \times (i + 0.5)$. En general, a mayor cantidad de puntos (n mayor) es *mejor* o *más precisa* la aproximación de π que se obtiene. Claramente, cada término de la sumatoria puede ser calculado de manera absolutamente independiente de todos los demás y, por lo tanto, estos cálculos (y las sumas intermedias de los mismos) pueden ser paralelizados en tantos procesadores como se decida. Si bien la paralelización es trivial, no sucede lo mismo con el balance de carga, dado que no se logra balancear la carga en p

procesadores realizando la suma parcial de *n*/*p* términos de la Ec. (1) en cada uno de ellos. En este caso particular, se puede aprovechar de manera directa otra de las características del cálculo planteado en la Ec. (1): todos los términos de la sumatoria implican el mismo costo en términos de operaciones numéricas. En este sentido, el problema no solamente es sencillo en cuanto a su división en partes sino que también es sencillo en cuanto a balance de carga en ambientes heterogéneos: la cantidad de términos de la sumatoria de la Ec. (1) a resolver en cada computadora es directamente proporcional a su potencia de cómputo relativa. En el caso de las PCs de la Tabla 1, la capacidad de cómputo relativa es tal que una es casi tres veces *mejor* que la otra en términos de capacidad de cálculo para π. Los resultados de los experimentos realizados con el cálculo de π se resumen en la Tabla 2, donde se muestran las velocidades relativas normalizadas con respecto a la de mayor capacidad de cómputo.

Velocidades Normalizadas	1 y 0,36
Eficiencia de la Paralelización	94%

Tabla 2: Resumen de la Experimentación con el Cálculo de π Interclusters.

Aunque la cantidad de computadoras es la mínima, el hecho de obtener una eficiencia del 94% es muy satisfactorio dado el bajo rendimiento de las interconexiones entre los clusters utilizados. En el caso de estas aplicaciones muy sencillas de paralelizar y balancear en cuanto a carga de trabajo, es de esperar que la eficiencia se mantenga alta aumentando la cantidad de computadoras.

5.2 Cálculos Asociados al Conjunto de Mandelbrot

La Fig. 4 muestra una de las formas más comunes de cómputo utilizado para el gráfico relacionado con el conjunto de Mandelbrot (mencionado en algunos casos como "*escape time algorithm*" [26]) para un punto dado como (x0, y0) [22].

```
x = x0; y = y0; iter = 0;
while ( x*x + y*y < (2*2)  AND  iter < max)
{
   xtemp = x*x - y*y + x0; y = 2*x*y + y0;
   x = xtemp; iter = iter + 1;
}
color = iter
```

Figura 4: Cómputo Relacionado con el Conjunto de Mandelbrot.

Comparándolo con el cómputo de π anterior, el que se muestra en la Fig. 4:

- Es similar en cuanto a que el valor de un punto en particular es totalmente independiente de todos los demás valores.
- Es diferente en cuanto a que la cantidad de operaciones necesarias para el cálculo del valor de un punto en particular no es conocida a priori, depende del punto mismo (iteración *while*).

Normalmente, lo que se paraleliza es el cálculo de los diferentes puntos y evidentemente no requiere mucho esfuerzo. No es tan directo el balance de carga, aún cuando se conozcan las velocidades relativas de las computadoras a utilizar. Tampoco es demasiado complejo, dado que se puede recurrir a un esquema similar al utilizado en cómputos de álgebra lineal, por ejemplo, donde se distribuye el cálculo de forma tal que cada proceso debe obtener valores relativamente dispersos en el espacio total a calcular [16]. En el caso de álgebra lineal, esto se aplica sobre la matriz a factorizar en L y U, por ejemplo, y en este caso se aplicará al conjunto de puntos a calcular. Visto como una matriz, el conjunto de puntos a calcular se puede dividir en múltiples bloques de relativamente pocas filas o columnas y estos bloques se asignan a las diferentes computadoras.

Teniendo una cantidad de bloques suficientemente grandes, la cantidad de bloques asignados a cada computadora es directamente proporcional a su velocidad relativa. En el caso específico del conjunto de Mandelbrot, se llevó a cabo el cálculo para un *espacio* de 800x800 puntos que se dividió en bloques de 100x800 puntos (bloques de 100 filas) y de cada uno de estos bloques, la cantidad de filas asignadas a cada computadora es proporcional a la velocidad relativa de las mismas. La Tabla 3 muestra el resumen de los experimentos realizados con este problema, aplicando el balance de carga que se describió previamente. Una vez más, la eficiencia de la paralelización es muy satisfactoria para la plataforma de cómputo subyacente.

Velocidades Normalizadas	1 y 0,36
Eficiencia de la Paralelización	89%

Tabla 2: Resumen de la Experimentación con el Cálculo de π Interclusters.

6 CONCLUSIONES Y TRABAJO FUTURO

En este artículo se ha mostrado la utilización de una VPN para cómputo paralelo interclusters utilizando una implementación específica de MPI. Esto muestra que, al menos en principio, no es necesaria una infraestructura muy compleja sino *middleware* estándar para cómputo paralelo interclusters. Los experimentos han mostrado que es posible caracterizar satisfactoriamente el rendimiento de las comunicaciones interclusters de manera sencilla, y a partir de esta información se pueden tomar decisiones importantes de paralelización. También se ha mostrado que no solamente es factible paralelizar aplicaciones, (al menos las del tipo *embarrassingly parallel*), sino que se puede obtener rendimiento muy satisfactorio, con eficiencia de la paralelización de alrededor del 90% o mayor. También se ha mostrado que las estrategias conocidas de balance de carga dan resultados satisfactorios, al menos en las aplicaciones que se mostraron.

Una de las extensiones inmediatas de este trabajo consiste en la utilización de más de una computadora en cada cluster involucrado. Sin lugar a dudas el objetivo final de cómputo paralelo interclusters debería ser el de utilizar todos los clusters de manera completa, para aprovechar al máximo la capacidad de cálculo disponible. En caso contrario, se debería establecer una estrategia o política de utilización de las computadoras de cada cluster que justifique el hecho de no utilizar las computadoras disponibles de los clusters.

REFERENCIAS

[1] E. Argollo, D. Rexachs, F. G. Tinetti, E. Luque, “Efficient Execution of Scientific Computation on Geographically Distributed Clusters”, Applied Parallel Computing: 7th International Conference, PARA 2004, Lyngby, Denmark, June 20-23, 2004, Revised Selected Papers, LNCS 3732. Editors: J. Dongarra, K. Madsen, J. Wasniewski, ISBN: 3-540-29067-2, Springer-Verlag Berlin Heidelberg 2006, pp. 691–698.
[2] W. Aróztegui, F. L. Romero, F. G. Tinetti, “Comunicaciones para Cómputo Paralelo Intercluster”, Anales del VIII Workshop de Investigadores en Ciencias de la Computación, Universidad de Morón, Facultad de Informática, Ciencias de la Comunicación y Técnicas Especiales, Morón, Argentina, Junio 1-2 de 2006, ISBN 950-9474-34-7, pp. 211-215.
[3] M. Baker, R. Buyya, “Cluster Computing at a Glance"”, en R. Buyya Ed., High Performance Cluster Computing: Architectures and Systems, Vol. 1, Prentice-Hall, Upper Saddle River, NJ, USA, pp. 3-47, 1999.
[4] G. Burns, R. Daoud, J. Vaigl, “LAM: An Open Cluster Environment for MPI”, Proc. of Supercomputing Symposium, pp. 379-386, 1994. Available at http://www.lammpi.org/download/files/lam-papers.tar.gz
[5] J. Dongarra, A. Geist, R. Manchek, V. Sunderam, “Integrated pvm framework supports heterogeneous network computing”, Computers in Physics, (7) 2, pp. 166-175, April 1993.
[6] G. Faag, J. Dongarra, A. Geist, “PVMPI provides interoperability between MPI implementations”, in

Proc. 8th SIAM Conf. on Parallel Processing, SIAM (1997).
[7] G. E. Fagg, K. S. London, "MPI interconnection and control", Technical Report Tech Rep. 98-42, Corps of Engineers Waterways Experiment Station Major Shared Resource Center (1998).
[8] I. Foster, "Globus Toolkit Version 4: Software for Service-Oriented Systems", IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2006.
[9] I. Foster, C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, 2nd Edition, Morgan Kaufmann, 2003, ISBN 1558609334.
[10] W. L. George, J. G. Hagedorn, J. E. Devaney, "IMPI: Making MPI Interoperable", J. of Research of the National Institute of Standards and Technology, Volume 105, Number 3, May–June 2000, pp. 343-348.
[11] W. Gropp, E. Lusk, "Sowing MPICH: A Case Study in the Dissemination of a Portable Environment for Parallel Scientific Computing", The International Journal of Supercomputer Applications and High Performance Computing, Vol. 11, No. 2, pp. 103-114, Summer 1997,
[12] T. Kielmann, H. E. Bal, S. Gorlatch, K. Verstoep, R. F.H. Hofman, "Network performance-aware collective communication for clustered wide-area systems", Parallel Computing archive, Volume 27, Issue 11, Oct. 2001, Clusters and computational grids for scientific computing, pp. 1431 – 1456, ISSN 0167-8191.
[13] T. Kielmann, R. F. H. Hofman, H. E. Bal, A. Plaat, R. A. F. Bhoedjang, "MAGPIE: MPI's collective communication operations for clustered wide area systems", Seventh ACMSIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'99), Atlanta, GA, May 1999, pp. 131–140.
[14] MPI Forum, "MPI: a message-passing interface standard", International Journal of Supercomputer Applications, 8 (3/4), pp. 165-416, 1994.
[15] A. Tanenbaum, "Structured Computer Organization", 5th Ed., Prentice Hall, ISBN 0131485210, 2005.
[16] Fernando G. Tinetti, Tesis Doctoral, Cómputo Paralelo en Redes Locales de Computadoras, Tesis Doctoral, Univ. Autónoma de Barcelona, 2004. Disponible en http://ftinetti.googlepages.com/tesisdoctoral
[17] Tinetti F. G., Aróztegui W., "Bibliotecas de Pasaje de Mensajes y Cómputo Intercluster", Rep. Técnico PLA-003-2005, Sep. 2005. Disponible http://ftinetti.googlepages.com/ reportest%C3%A9cnicos2006
[18] F. G. Tinetti, W. J. Aróztegui, "Factibilidad y Rendimiento de las Comunicaciones para Cómputo Paralelo Intercluster", XII Congreso Argentino de Cencias de la Computación (CACIC), Oct. 17-21, 2006, Universidad Nacional de San Luis, Potrero de los Funes, San Luis, Argentina.
[19] F. G. Tinetti, W. Aróztegui, "Instalación y Configuración de OpenVPN 2.0 para Cómputo Paralelo Intercluster", Reporte Técnico PLA-002-2006, Julio 2006. Disponible en http://ftinetti.googlepages.com/ reportest%C3%A9cnicos2006
[20] F. G. Tinetti, W. Aróztegui, "Perfil Preliminar de las Comunicaciones Intercluster", Reporte Técnico PLA-001-2006, Marzo 2006. Disponible en http://ftinetti.googlepages.com/reportest%C3%A9cnicos2006
[21] F. G. Tinetti, A. Quijano, A. De Giusti, E. Luque, "Heterogeneous Networks of Workstations and the Parallel Matrix Multiplication", Recent Advances in Parallel Virtual Machine and Message Passing Interface, 8th European PVM/MPI, Santorini/Thera, Greece, Sep. 23-26, 2001, Proceedings, Y. Cotronis, J. Dongarra (Eds.). LNCS 2131 Springer 2001, ISBN 3-540-42609-4, pp. 296-303.
[22] B. Wilkinson, M. Allen, Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, 2nd Ed., Prentice Hall, 2004, ISBN 0131405632.
[23] J. Yonan, "Understanding the User-Space VPN: History, Conceptual Foundations, and Practical Usage", Linux Fest Northwest 2004. Disponible en http://openvpn.net/papers/BLUG-talk/BLUG-talk.ppt
[24] Albatross: Wide Area Cluster Computing Homepage, http://www.cs.vu.nl/albatross/
[DAS-3] The Distributed ASCI Supercomputer 3 (DAS-3) Homepage, http://www.cs.vu.nl/das3/
[25] Wikipedia, the free encyclopedia, "Virtual Private Networks", http://en.wikipedia.org/wiki/Virtual_ private_network
[26] Wikipedia, the free encyclopedia, "Mandelbrot set", http://en.wikipedia.org/wiki/Mandelbrot_set