Bibliotecas de Pasaje de Mensajes y Cómputo Intercluster

Fernando G. Tinetti* Walter Aróztegui

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

Reporte Técnico PLA-003-2005[1]
Septiembre 2005

Resumen

Este reporte está orientado básicamente a documentar el estudio realizado de las bibliotecas más conocidas de pasaje de mensajes en el contexto de su posible utilización para cómputo paralelo intercluster. En un reporte técnico anterior, se describía lo más importante del servicio de shell seguro (*secure shell*) que se asume como el más sostenible desde el punto de vista técnico para la interconexión de dos o más clusters con la posibilidad de *login* remoto y disparo de tareas remotas. En este reporte se enfocan las bibliotecas de pasaje de mensajes específicamente en lo referente a la sostenibilidad de las características de seguridad. Desde esta perspectiva de seguridad, como en el caso del servicio ssh, se debe tener en cuenta como mínimo la protección provista por los firewalls de cada cluster. Desde el punto de vista del rendimiento, por su parte, se debe tener en cuenta que para cómputo intercluster normalmente existen fuertes diferencias de rendimiento de las redes de interconexión involucradas.

1. Introducción

A partir de la introducción de TCP/IP [8] la interconexión de dos redes locales ha sido una tarea más o menos rutinaria utilizando *routers*. La Fig. 1 muestra la visión clásica en este contexto, donde cada red

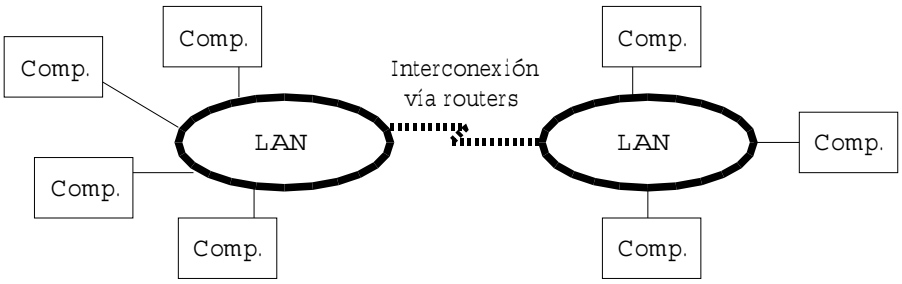

Figura 1: Interconexión de Dos Redes Locales.

local dispone de un router para llevar a cabo la transmisión de datos entre ambas redes locales. Por otro lado, la Fig. 2 muestra la visión clásica de cómputo paralelo en un entorno MIMD (Multiple Instruction stream, Multiple Data stream) [2] [3] con pasaje de mensajes. Esta visión (Fig. 2) es casi directamente la utilizada/adoptada por la mayoría de las bibliotecas de cómputo paralelo vía pasaje de mensajes en clusters [6] [1] [7] [4] [5]. En realidad, estas bibliotecas crean un entorno de programación y ejecución de programas paralelos sobre clusters ocultando todos los detalles propios de cada computadora (arquitectura e interfase de comunicaciones, por ejemplo) y proveyendo identificación única de procesos más las rutinas propias de pasaje de mensajes. Desde la perspectiva de los procesos de un programa paralelo que utilizan una biblioteca de pasaje de mensajes, la situación es la que se muestra esquemáticamente en la Fig. 3. En un entorno estándar de cómputo paralelo en un cluster, las herramientas también estándares son la utilización de alguna biblioteca de pasaje de mensajes tal como una implementación de MPI junto con

* Investigador Asistente CICPBA
[1] PLA: sigla de Parallel Linear Algebra

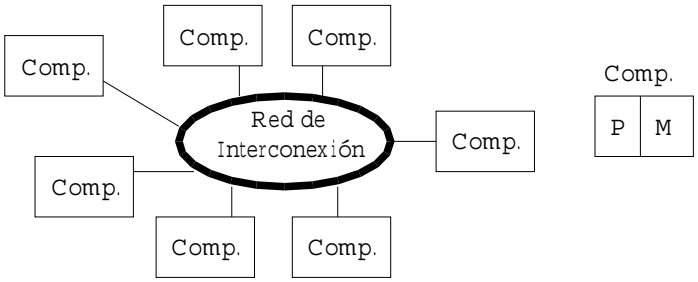

Figura 2: MIMD de Memoria Distribuida.

los servicios básicos de disparo de procesos remotos: rsh. De hecho, cualquier instalación estándar de las implementaciones de MPI tales como MPICH y LAM/MPI utiliza el comando `mpirun` como herramienta para la ejecución de programas paralelos SPMD (Single program, Multiple Data) la cual, a su vez, utiliza el servicio rsh para el disparo de comandos en otras máquinas (o *máquinas remotas*). En un cluster utilizado para cómputo paralelo se asume que no hay problemas *internos* de seguridad, sino que los problemas de seguridad (si existen) provienen del exterior, en el caso de que el cluster esté conectado a Internet, por ejemplo [9]. Sin embargo, cuando se vuelve al contexto de cómputo paralelo interclusters, los servicios

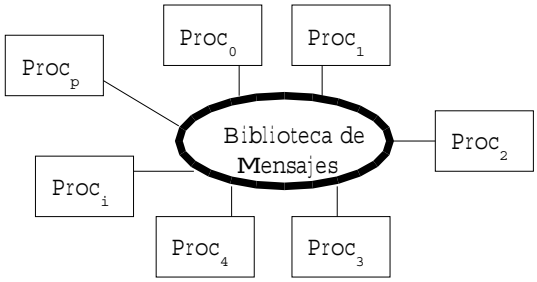

Figura 3: Programa con Pasaje de Mensajes.

de disparo remoto no representan los únicos problemas de seguridad. Más específicamente, no solamente se deben disparar tareas remotas sino que, además, los procesos se comunican en tiempo de ejecución. De acuerdo con las implementaciones de las bibliotecas de pasaje de mensajes, estas comunicaciones se implementan usando los protocolos TCP y/o UDP sobre IP desde la interfase socket. En términos de seguridad y relacionado con los posibles firewalls de cada cluster, esto implica la comunicación utilizando posiblemente múltiples ports, normalmente *no privilegiados* (*unpriviledged*). Resumiendo, es de esperar que para el disparo de tareas remotas los entornos de ejecución con pasaje de mensajes utilicen algún servicio de rsh o ssh y para las transferencias de datos entre los procesos utilicen una o más conexiones TCP y/o UDP.

Teniendo en cuenta las consideraciones anteriores y enfocando específicamente los problemas de seguridad y los controles impuestos vía firewalls, se deben identificar los puertos que utilizan las biliotecas de pasaje de mensajes y sus entornos de ejecución. Si bien es muy dificil identificar claramente todos los detalles de las comunicaciones entre las computadoras, dado que se depende de las implementaciones (versiones, por ejemplo, de PVM o de las implementaciones de MPI), al menos se considera útil la identificación de cantidades y *tipos* de puertos (al menos en términos de privilegiados o no) a utilizar desde los programas paralelos con pasaje de mensajes. En este sentido, se presenta a continuación un estudio relativamente detallado del funcionamiento de la versión LAM/MPI en la ejecución de programas muy sencillos, con tres objetivos:

- Tener una implementación real de referencia. Si bien siempre el alcance será limitado a la implementación misma, también es cierto que se tienen todos los detalles implementados y en uso. Por otro lado, LAM/MPI es una de las dos implementaciones de uso libre de MPI, junto con MPICH.
- Aunque las extrapolaciones son relativamente complejas, sí es posible identificar similitudes y diferencias de LAM/MPI con el resto de las bibliotecas de pasaje de mensajes de uso libre disponibles

actualmente: PVM y MPICH. En la medida de lo posible, estas similitudes y diferencias se irán detallando junto con el análisis de los resultados obtenidos de la experimentación.

- Sin lugar a dudas, el objetivo final es la factibilidad técnica de utilizar una biblioteca de pasaje de mensajes para cómputo intercluster. En el mejor de los casos de programas y contexto de ejecución de los mismos, la idea sería utilizar directamente la biblioteca como si todas las computadoras estuvieran en un único cluster y concentrar el esfuerzo en los problemas de rendimiento involucrados por la comunicación con máquinas *locales* y *remotas*.

En cualquier caso, los detalles que se muestran a continuación de la ejecución con LAM/MPI también darán idea de algunas decisiones de diseño del entorno de ejecución que no necesariamente son directamente relacionados con las rutinas de pasaje de mensajes definidas en el estándar MPI.

2. Breve Descripción de LAM/MPI

LAM/MPI (Local Area Multicomputer/Message Passing Interface) [1] es un ambiente de programación y un sistema de desarrollo MPI para computadoras posiblemente heterogéneas en una red de trabajo, de manera que cualquier infraestructura de computación en red existente puede transformarse en un sistema de cómputo paralelo. LAM/MPI provee a los usuarios no solamente con el API (Application Programming Interface) MPI estándar, es decir una implementación completa del estándar MPI-1 y algunos elementos de MPI-2, sino también con varias herramientas de monitoreo y depuración (debugging). Si bien está específicamente orientada a clusters posiblemente heterogéneos de estaciones de trabajo (workstations) UNIX, es capaz de correr sobre una amplia gama de plataformas, desde las PC's de escritorio hasta grandes supercomputadoras (y todo lo que haya entre estos extremos), incluyendo arquitecturas de 64 bits.

2.1. Instalación y configuración de LAM

La distribución LAM/MPI 6.5.6 que se ha utilizado en los experimentos, está empaquetada en un archivo comprimido, lam-6.5.6.tar.Z o lam-6.5.6.tar.gz, que puede ser obtenido directamente desde el sitio web de LAM/MPI [10] pues es de distribución gratuita. Es posible que para algunas versiones deban obtenerse también los llamados parches (patches), que cuentan con soluciones para algunos problemas observados por los usuarios.

Para descomprimir el archivo y extraer el código fuente se puede ejecutar en la línea de comandos linux:

> gunzip -c lam-6.5.6.tar.gz | tar xf

o

> uncompress -c lam-6.5.6.tar.Z

según sea el caso. Esto creará un directorio llamado lam-6.5.6 con todo el código de LAM/MPI desempaquetado adentro. LAM/MPI usa un script de GNU para una adecuada configuración pues son necesarias diferentes compilaciones para cada versión distinta de sistema operativo así como para cada arquitectura. Por ejemplo, si se tienen versiones de Solaris 2.5 y 2.6, se aconseja que se debe correr el script de configuración para cada una, aunque en algunos casos como distintas versiones de Linux Red Hat, la experiencia indica que es suficiente la instalación y configuración completa en una sola máquina y luego copiar los archivos (manteniendo la misma estructura) a las demás computadoras que constituyen la red a usar. La configuración se realiza ejecutando el script en el directorio de LAM:

> ./configure options

o

> sh ./configure options

Son muchas las opciones que pueden ser cambiadas con este script antes de la instalación, pues la configuración debe considerar un amplio abanico de características que van desde el directorio donde se instala hasta qué tipo de comunicación o compilador se utilizará con LAM/MPI. Algunas de ellas se verán a continuación.

a) Directorio de instalación: a menos que se indique lo contrario, LAM/MPI se instala dentro en /usr/local (que debe tener permiso de escritura porque de lo contrario la instalación fallará), para cambiar esto, se debe utilizar la opción

--prefix

en configure, así para que los archivos binarios se instalen en, por ejemplo, $HOME/LAM, habrá que hacer:

> cd lam-6.5.6

> ./ configure --prefix=$HOME/LAM

con esto, luego de la instalación se tendrán los archivos binarios en $HOME/LAM/bin

b) Compiladores: LAM/MPI no requiere la instalación de compiladores C++ ni Fortran para trabajar apropiadamente, por lo que se puede especificar que no se instalen las interfaces para esos lenguajes con la opción de configuración -without-...:

> cd lam-6.5.6

> ./ configure -without-cxx (para C++)

o

> ./ configure -without-fc (para Fortran)

con lo que luego de la instalación no se podrán usar las interfaces de MPI2 C++ ni de MPI Fortran que están incluídas en el paquete de LAM/MPI.

Se pueden hacer también opciones combinadas, por ejemplo si se quieren utilizar compiladores/interfases para C y C++, pero no para Fortran, la configuración se hará con:

> cd lam-6.5.6

> ./ configure --with-cc=cc --with-cxx=CC -without-fc

c) Flags del linker: de la misma manera que con los compiladores, se pueden elegir en la configuración las opciones del linker, por ejemplo si se quiere activar el soporte para la depuración (debugging) para todos los ejecutables LAM/MPI:

> cd lam-6.5.6

> ./ configure -with-ldflags=-g

d) Mecanismos de pasaje de mensajes: LAM/MPI ofrece tres diferentes RPI's (Request Progression Interface) que son las formas en que un mensaje se transmite desde un proceso fuente a un poceso destino:

- tcp (opción usada a menos que se indique otra) que usa TCP/IP para todas las comunicaciones MPI.
- usysv usa memoria compartida para comunicaciones MPI dentro de un mismo nodo y TCP/IP para las comunicaciones "extra-nodo".
- sysv igual que usysv excepto algunas diferencias en el manejo de la memoria compartida (utilizando semáforos).

Si se quiere cambiar la configuración de comunicaciones, se utilizará la opción --with-rpi. Por ejemplo, para especificar usysv se utilizará:

> cd lam-6.5.6

> ./ configure --with-rpi=usysv

e) Programa de shell remoto (remote shell): es necesario contar con un apropiado programa de shell remoto, pues LAM/MPI debe ser capaz de ejecutar comandos en otras máquinas. En la configuración se puede cambiar el que se utiliza de manera estándar (algunas veces suele estar no disponible por razones de seguridad) con la opción --with-rsh, entonces para especificar por ejemplo el programa secure shell (ssh) se usará:

> cd lam-6.5.6

> ./ configure --with-rsh=/bin/ssh -x"

donde -x es una de las opciones de ssh.

Son muchas más las opciones con las que modificar la configuración estándar, pero es conveniente consultar el manual de LAM/MPI en cada caso específico. Una vez que se han elegido las alternativas de configuración, sólo queda ejecutar el comando

> make

y LAM/MPI se compilará e instalará en el directorio especificado y con todas las opciones seleccionadas. También si se invoca el comando

> make examples

desde el directorio de LAM/MPI, se instalarán una serie de ejemplos que suelen ser de mucha ayuda al construir los programas de aplicación. Para utilizar un cluster, LAM/MPI debe estar instalado en todos

los nodos. Si bien como se explica antes, sobre distintas versiones de Linux Red Hat, LAM/MPI funciona bien copiando los archivos desde una de las máquinas donde fue instalado, es imprescindible que la versión de LAM/MPI sea la misma en todos los procesadores.

2.2. Configuración de usuario

LAM/MPI fue diseñado para ser corrido por usuarios individuales y no para correr como un servicio a nivel root donde múltiples usuarios acceden a los mismos servidores LAM/MPI de modo cliente/servidor. Aunque sí puede ser cargado por cada usuario que desee utilizar programas LAM/MPI y funcionar a la vez en la misma red, pero con la resultante caída en el rendimiento.

De esto surge la necesidad de que cada usuario debe tener una cuenta en todas las máquinas donde pretende utilizar LAM/MPI y configurar adecuadamente el entorno (environment), por lo tanto necesita que se cumplan principalmente dos cosas:

1. El directorio para los ejecutables LAM/MPI deberá estar especificado en el path del usuario en todas las máquinas que se van a usar. Típicamente esto se define para cada usuario en un archivo de inicialización como $HOME/.cshrc, $HOME/.profile, $HOME/.bashrc u otros de acuerdo al shell que se utilice.

2. El usuario necesita ser capaz de ejecutar comandos remotos sin tener que introducir una clave al acceder a otras máquinas, y sin salidas extrañas en stderr. LAM/MPI usa rsh si no se ha cambiado esto explícitamente en la configuración o a través de una variable de entorno (típicamente LAMRSH). Para prescindir de la clave en cada acceso, es necesario que el usuario tenga un archivo $HOME/.rhosts, usualmente con permisos 644 (rw-------) y en el que figura una lista de máquinas (hosts) y usuarios que tienen permitido ejecutar comandos o lanzar programas en la máquina sobre la que está el archivo .rhosts. Si, por ejemplo, el usuario usr1 desde la máquina llamada uno necesita lanzar programas en las máquinas dos, tres y cuatro, en cada una de éstas debe haber un archivo .rhosts con una línea:

    uno usr1

si rsh no encuentra esta línea pedirá el ingreso del password, y en estas condiciones el entorno de LAM/MPI no se podrá establecer. Además, es posible que la computadora y/o el usuario deba estar asentado en una tabla de confianza (generalmente /etc/host.equiv) en todas las máquinas a utilizar con LAM/MPI y en el que figura el nombre del usuario y las máquinas desde las que tiene permiso acceder.

2.3. Carga de LAM/MPI, compilación y ejecución de programas

Una vez configuradas adecuadamente las cuentas de los usuarios que van a utilizarlo, con los debidos permisos, paths, variables de entorno, etc., es posible empezar a utilizar LAM/MPI. El ambiente LAM/MPI necesita ser cargado antes de ejecutar aplicaciones, LAM/MPI utiliza procesos sobre cada nodo para el control de los procesos de usuario (programa paralelo en ejecución). El comando

    > lamboot lamhosts

lanza estos "procesos LAM" sobre cada una de las máquinas que conformarán el cluster. En cierta forma, LAM/MPI tiene la idea de "máquina virtual paralela" de PVM y este comando lamboot es equivalente a abrir la consola de PVM con el conjunto de hosts a utilizar para las aplicaciones paralelas. El archivo lamhosts es una lista de inicialización donde se incluyen las máquinas que van a intervenir en el ambiente LAM/MPI y puede estar formado por máquinas individuales o por nodos múltiples (un nodo puede estar formado por varios CPU's, como en el caso de utilizar sistemas SMPs), la lista puede estar formada por:

    maquina1
    maquina2
    maquina3
    maquina3
    maquina4 cpu=3

Se ven en el ejemplo dos formas de indicar nodos múltiples, una con la repetición en la lista del nombre del nodo (maquina3 tendrá 2 CPU's), y la otra manera es agregando la frase "cpu=N"donde N es la cantidad de CPU's que componen el nodo. Con el entorno LAM/MPI ya corriendo, esta lista puede ser modificada agregando o quitando nodos al entorno con comandos como

    > lamgrow

y

    > lamshrink

respectivamente y los nodos accesibles a LAM/MPI pueden verificarse con la herramienta recon. El entorno LAM/MPI provee compiladores del tipo "wrapper" como mpicc o hcc, mpiCC, o hcp y mpif77 o hf77 para compilar programas MPI en C, C++ y Fortran. La creación de un programa LAM/MPI

requiere links a bibliotecas específicas del entorno que pueden no residir en uno de los directorios de búsqueda estándar y también es frecuente la inclusión de archivos cabecera que pueden no encontrarse en las ubicaciones corrientes. El compilador de LAM/MPI pasa los argumentos al compilador nativo de C junto con las opciones -I, -L y -l requeridas por los programas. De aquí que las opciones de compilación sean las mismas que éste. Las opciones pueden verse invocando por ejemplo el comando:

> hcc -showme

La ejecución se hace con el comando mpirun, comando muy potente que cuenta con varias opciones que van desde configurar en cuántas y cuáles máquinas cargadas en el entorno LAM/MPI se debe ejecutar el programa, tales como

C: ejecutar 1 proceso en cada uno de los nodos del entorno,

-c (o -np) n: ejecutar n procesos en todos los nodos (n puede ser mayor o menor que la cantidad de máquinas),

-np3-6 : ejecutar en los nodos 3 a 6,

hasta opciones que delinean el tipo de comunicación a usar en los procesos (conviene referirse al manual de LAM/MPI para conocer cada una de ellas). Cuando mpirun tiene un nombre de archivo relativo, por ejemplo

> mpirun C programa

trata de encontrar la aplicación en su $PATH sobre todos los nodos a ejecutar, siguiendo el modelo de ejecución del shell UNIX, si en cambio el nombre del archivo es absoluto, tal como

> mpirun C $HOME/programa

simplemente trata de ejecutar ese nombre absoluto, sobre todos los nodos. Otra cosa que permite LAM/MPI es la ejecución de varios programas a la vez e inclusive determinar para ello nodos distintos, como por ejemplo:

> mpirun c0-3 programa1

> mpirun c4-7 programa2

o correr programas de acuerdo a un esquema de ejecución previamente cargado en un script. LAM/MPI tiene muchas opciones de funciones y comandos, no es posible explicar aquí todas estas capacidades. Para un conocimiento más exhaustivo, sin duda habrá que referirse al manual.

2.4. LAM/MPI funcionando con ssh

Como se ha mencionado anteriormente, el agente de disparo remoto usado por LAM/MPI inicialmente es rsh, pero puede cambiarse a otros agentes como ssh, que incrementan la seguridad sobre las autenticaciones por .rhosts de rsh y permite la autenticación a través de pasaje de token AFS (altamente segura). Se puede especificar la utilización de ssh, en el momento de configurar antes de instalar las bibliotecas y comandos, con el comando configure, utilizando la opción:

> ./configure --with-rsh="ssh -x"

donde -x previene mensajes de status de xauth impresos sobre stderr y que lamboot o recon pueden interpretar como una falla de una invocación remota, abortando la formación del cluster. Otra manera es fijando la variable de entorno, antes de invocar los comandos recon, lamboot o cualquier otro ejecutable LAM/MPI que fuerce a LAM/MPI a usar el shell remoto. Por ejemplo, usando un shell Bourne:

> LAMRSH="ssh -x"

> export LAMRSH

> recon myhostfile

y por lo tanto también debe funcionar fijando previamente esta variable (LAMRSH) en el profile del shell (.bash_profile, por ejemplo).

2.5. Distintos Modos de Pasaje de Mensajes en LAM/MPI

Hay dos maneras en que LAM/MPI puede transferir mensajes entre procesos de la aplicación paralela: modo "proceso LAM/MPI" o *daemon* (denominado lamd a partir de aquí) y modo "cliente a cliente" (denominado C2C a partir de aquí):

- En modo LAMD, todos los mensajes MPI son pasados entre procesos vía los procesos LAM/MPI que son lanzados con el lamboot al iniciar el cluster. Esto significa que para pasar mensajes entre dos procesos a y b, ejecutándose en las computadoras A y B respectivamente se sigue la ruta:

$$\text{Proceso a} \longrightarrow \text{Proceso LAM/MPI en A} \longrightarrow \text{Proceso LAM/MPI en B} \longrightarrow \text{Proceso b}$$

Donde se tienen tres saltos hasta llegar al proceso destino, salvo que los procesos a y b se ejecuten en el mismo nodo, por lo que usarían el mismo proceso LAM/MPI y habría sólo dos saltos.

- En modo C2C, los mensajes desde el proceso a hasta el proceso b siguen directamente la ruta:

  $$\text{Proceso a} \longrightarrow \text{Proceso b}$$

  Ahora los procesos LAM/MPI no se involucran y por lo tanto hay un solo salto desde un proceso a otro, que involucra la red de comunicaciones si se ejecutan en diferentes computadoras.

El modo lamd es generalmente más lento que el C2C, por la mayor cantidad de saltos necesarios, pero tiene algunas ventajas:

- Permite la utilización de terceras aplicaciones como XMPI, posibilitando el monitoreo de pasaje de mensajes y creación de reportes de desempeño de los programas MPI.
- Puede usarse el comando mpimsg que proporciona instantáneas del programa mpi que se está ejecutando.
- Tiene una significativa mayor capacidad de buffering que el C2C. Todos los mensajes son pasados vía un socket UNIX desde el programa MPI al proceso LAM/MPI, puediendo éste almacenar en buffer una gran cantidad de mensajes de entrada y salida. La capacidad de buffer de C2C es mucho menor y algunos programas pueden llegar a tener deadlock cuando se llena el buffer.
- Los procesos lamd pueden realizar comunicaciones "no-bloqueantes" de manera más eficiente, pues pueden enviar mensajes en "background" con respecto al programa de usuario.

A menos que se indique explícitamente lo contrario, LAM/MPI funciona con el modo lamd, pero esto puede cambiarse a través de la opción del comando mpirun (ver manual):
> mpirun -c2c C myprogram

3. Detalle de Experimentos con LAM/MPI

La idea inicial de la experimentación con LAM/MPI ha sido la de identificar las características de conexiones entre procesos con el objetivo de evaluar la sustentabilidad técnica de interconectar dos clusters para cómputo paralelo. Se debe tener en cuenta que esta sustentabilidad está directamente relacionada con la posibilidad de la existencia de restricciones en cuanto a las interconexiones que imponen los firewalls por razones de seguridad. Es de destacar que, aunque los firewalls de protección de cada cluster pueden ser modificados, normalmente no es posible modificar todos los firewalls que posiblemente existen en los routers intermedios. Básicamente, se debe identificar el tipo y la cantidad de conexiones que se utilizan para ejecutar aplicaciones de cómputo paralelo con bibliotecas de pasaje de mensajes. En este caso en particular, se utiliza LAM/MPI como una de las posibles y además como representativa de las bibliotecas de uso libre disponibles.

Normalmente, las conexiones entre procesos se implementan usando sockets con protocolo TCP y/o UDP entre las máquinas y/o directamente entre los procesos de la aplicación paralela. Esta es una de las características comunes a las bibliotecas LAM/MPI, MPICH y PVM. Para la visualización de los puertos se utilizaron distintos analizadores de red, corriendo aplicaciones LAM/MPI en diferentes circunstancias muy simples. En cuanto a la cantidad de máquinas, se probaron aplicaciones que usan dos y tres computadoras. En cuanto a los programas se utilizan solamente las funciones Send-Recv. En cuanto a la cantidad de datos, se utilizaron dos tamaños de mensajes: 100 y 1000 enteros. En el caso particular de LAM/MPI, se aprovecha la posibilidad de iniciar la *máquina paralela*, es decir iniciar LAM/MPI pero sin ejecutar ningún programa paralelo propiamente dicho y también se probaron los modos de ejecución lamd y C2C para la transferencia de datos entre los procesos. Cada una de estas decisiones tiene en cuenta:

- Con las diferentes cantidades de máquinas (dos y tres) se intenta identificar si existe alguna relación entre la cantidad de computadoras y la cantidad de conexiones a abrir. En el caso de los programas paralelos, se asigna un proceso de usuario en cada computadora.
- Se utilizan solamente las funciones Send-Recv porque son las más sencillas que pueden tener los programas paralelos. Se descartaron en esta instancia las operaciones más complejas como las de comunicaciones colectivas para no hacer muy complejo el análisis de resultados.

- Los tamaños de mensajes de 100 y 1000 enteros solamente se usaron para identificar las conexiones a través de las cuales se transfieren los datos, en el caso de que haya más de una conexión entre las computadoras.
- Se analiza el inicio LAM/MPI sin ejecutar programas para verificar si el propio ambiente de ejecución de LAM/MPI establece conexiones independientes de los programas paralelos a ejecutar.
- Los distintos modos de transferencia de datos entre los procesos se probaron para identificar si estas transferencias se traducen en distintas conexiones TCP y/o UDP entre las máquinas.

En la sección siguiente se describen los resultados obtenidos y también las conclusiones de cada uno de los experimentos en cuanto a la identificación de las conexiones entre las computadoras involucradas.

4. Análisis de Resultados Obtenidos

Los primeros resultados corresponden a la inicialización de LAM/MPI sin ejecutar ningún programa paralelo, es decir con el comando lamboot de LAM/MPI. Los puertos utilizados durante la ejecución del comando lamboot en la máquina A y que involucra iniciar LAM/MPI en la propia máquina A y en la máquina B se muestran en la Tabla 1. En cierta forma, los datos de la Tabla 1 son bastante predecibles,

Puerto A	Puerto B	Protocolo
32843	22 (ssh)	TCP
32844	22 (ssh)	TCP
32840	32771	TCP
32846	32770	TCP

Tabla 1: Puertos por Máquina en lamboot (Dos Computadoras).

dado que inicialmente se establece la conexión desde la máquina A (donde se ejecuta lamboot) hacia la máquina B. Desde la máquina A se inicia una coexión como cliente (puerto/s no protegido/s) con el servidor de ssh (puerto protegido 22) en la máquina B. Es razonable que a partir de la conexión cliente/servidor quede una conexión entre puertos no protegidos, pero sin embargo aparentemente hay dos pares de puertos no protegidos conectados: 32840 (A) con 32771 (B) y 32846 (A) con 32770 (B). Con el fin de constatar las conexiones que quedan abiertas post-lamboot, se ejecuta luego en cada máquina

```
> netstat -atup
```

que muestra todos los puertos conectados y los procesos a los que pertenecen. La Tabla 2 muestra los puertos abiertos en cada máquina relacionados con LAM/MPI después de haber ejecutado el comando lamboot. Es súmamente interesante notar que hay dos puertos en cada máquina y ambos pertenecientes

Puerto (máq.)	Proceso	Protocolo
32775 (A)	lamd	UDP
32776 (A)	lamd	UDP
32770 (B)	lamd	UDP
32771 (B)	lamd	UDP

Tabla 2: Conexiones Post-lamboot (Dos Computadoras).

al proceso lamd, que es el único involucrado en el comando lamboot. Con la intención de verificar si hay siempre dos puertos por máquina o la cantidad de puertos es dependiente de la cantidad de computadoras involcradas en lamboot, se llevaron a cabo los mismos experimentos y comprobaciones pero con tres máquinas: A, B y C. Los puertos utilizados durante la ejecución del comando lamboot en la máquina A y que involucra iniciar LAM/MPI en la propia máquina A y en las máquinas B y C se muestran en la Tabla 3. Los resultados del mismo comando pero para dos máquinas en cierta forma se replican para tres. Es decir: hay dos puertos no protegidos (*cliente*) que se conectan con el servidor de ssh (puerto 22) y luego hay una conexión entre puertos no protegidos desde ambos lados (*cliente* y *servidor*). Al ejecutar netstat en las tres máquinas involucradas, los resultados son los que se muestran en la Tabla 4. Estos últimos resultados muestran claramente que la cantidad de puertos por máquina es independiente de la cantidad de computadoras que se utilizan para cómputo paralelo. Más específicamente, hay dos puertos

Puerto (máq.)	Puerto (máq.)	Protocolo
32791 (A)	22 (ssh) (B)	TCP
32792 (A)	22 (ssh) (B)	TCP
32788 (A)	32771 (B)	TCP
32793 (A)	22 (ssh) (C)	TCP
32794 (A)	22 (ssh) (C)	TCP
32788 (A)	32771 (C)	TCP
32796 (A)	32770 (B)	TCP
32797 (A)	32770 (C)	TCP

Tabla 3: Puertos por Máquina en lamboot (Tres Computadoras).

UDP por máquina independientemente de que se utilice un cluster de dos o tres o cualquier cantidad de computadoras para cómputo paralelo. Es muy interesante notar la similitud entre el comando lamboot y el

Puerto (máq.)	Proceso	Protocolo
32771 (A)	lamd	UDP
32772 (A)	lamd	UDP
32770 (B)	lamd	UDP
32771 (B)	lamd	UDP
32770 (C)	lamd	UDP
32771 (C)	lamd	UDP

Tabla 4: Conexiones Post-lamboot (Tres Computadoras).

comando de inicio de PVM (pvm). En el caso de PVM también se inicia un proceso en cada computadora (pvmd, en este caso) que es el encargado de mantener la información y la consistencia local de la máquina paralela formada por el cluster de computadoras. También en el caso de PVM, la cantidad de puertos por máquina es constante (independiente de la cantidad total de máquinas) y, más específicamente, es un único puerto UDP.

Con respecto a la ejecución de programas paralelos y su relación con las conexiones entre computadoras, se ejecutó el programa con dos procesos que solamente utilizan un único par Send-Recv para la transferencia de 100 enteros desde una computadora hacia la otra en modo lamd. La Tabla 5 muestra los puertos utilizados en cada máquina al ejecutar el comando mpirun correspondiente. Es decir que se

Puerto (máq.)	Puerto (máq.)	Protocolo
32775 (A)	32771 (B)	UDP
32776 (A)	32770 (B)	UDP
32849 (A)	32774 (B)	TCP

Tabla 5: Puertos por Máquina para mpirun (Send-Recv) en Dos Computadoras.

utilizaron los dos puertos UDP previos al comando mpirun (Tabla 2), y aparece una conexión TCP más entre ambas computadoras que es la que lleva a cabo la transferencia de los datos. Al ejecutar el comando

> lamhalt

inmediatamente después del mpirun, los puertos utilizados son los que se muestran en la Tabla 6. Teniendo en cuenta los datos que se muestran en la Tabla 1 del comando lamboot para dos computadoras, los datos de la Tabla 2 que muestra los puertos abiertos después del comando lamboot y la pertenencia de estos puertos al proceso lamd, los datos de la Tabla 5 que muestra los puertos utilizados para el comand mpirun de un programa paralelo sencillo y los datos de la Tabla 6 que muestra los puertos utilizados al ejecutar el comando lamhalt, es claro que hay dos puertos UDP que se utilizan para comunicar información de control *entre* los procesos lamd. Dependiendo de lo que haya que ejecutar en la máquina paralela construida por LAM/MPI, se utilizan otros puertos/conexiones.

La Tabla 7 muestra los puertos utilizados para la ejecución del comando mpirun en tres computadoras de un programa sencillo (Send-Recv) que transfiere 100 enteros entre las computadoras en modo lamd. Básicamente, se utilizan los puertos preexistentes más un puerto TCP más por cada máquina que se

Puerto (máq.)	Puerto (máq.)	Protocolo
32775 (A)	32771 (B)	UDP
32776 (A)	32770 (B)	UDP

Tabla 6: Puertos por Máquina para lamhalt en Dos Computadoras.

Puerto (máq.)	Puerto (máq.)	Protocolo
32771 (A)	32771 (B)	UDP
32772 (A)	32770 (B)	UDP
32771 (A)	32771 (C)	UDP
32772 (A)	32770 (C)	UDP
32805 (A)	32783 (B)	TCP
32805 (A)	32788 (C)	TCP

Tabla 7: Puertos por Máquina para mpirun (Send-Recv) en Tres Computadoras, Modo lamd.

comunica en el programa paralelo. El comportamiento en cuanto a la utilización de puertos del mismo programa en las mismas máquinas pero con modo C2C se muestra en la Tabla 8. Es como si nada hubiera

Puerto (máq.)	Puerto (máq.)	Protocolo
32771 (A)	32771 (B)	UDP
32772 (A)	32770 (B)	UDP
32771 (A)	32771 (C)	UDP
32772 (A)	32770 (C)	UDP
32806 (A)	32786 (B)	TCP
32806 (A)	32792 (C)	TCP

Tabla 8: Puertos por Máquina para mpirun (Send-Recv) en Tres Computadoras, Modo C2C.

cambiado respecto del caso anterior. Esto puede significar que la implementación decide en tiempo de ejecución qué tipo de conexiones utiliza de acuerdo a la cantidad, tipo y tamaño de los mensajes que envían/reciben los procesos de la aplicación de usuario. Los experimentos con tamaños de mensajes mayores y con dos computadoras dan resultados similares entre sí y similares a los que se mostraron previamente. Quizás sería interesante analizar mejor las diferencias en cuanto a los diferentes modos de transferencia de mensajes, pero eso estaría fuera del análisis de lo que corresponde a los puertos/conexiones de LAM/MPI y su relación con la sostenibilidad técnica de usar directamente una biblioteca de pasaje de mensajes en el contexto de cómputo intercluster.

En resumen, se podría contabilizar la apertura de puertos por parte de la máquina que lanza las aplicaciones, de la siguiente manera:

- lamboot:
  - 3 puertos TCP de negociación con ssh y uno más al lanzar los lamd por cada máquina a comunicar (con 3 máquinas en total, se tienen 2 máquinas con las que comunicar).
  - 2 puertos UDP por cada máquina a comunicar, que permanecerán abiertos hasta el lamhalt.

  Total de puertos utilizados durante lamboot: 6 puertos por máquina.
- mpirun (c2c/lamd): sin contabilizar los puertos que se mantuvieron abiertos desde el lamboot, se abre 1 puerto TCP por cada máquina a comunicar.
- lamhalt: no abre puertos adicionales, solamente cierra los que utilizaban los lamd.

Es también importante notar que los únicos puertos privilegiados utilizados son los que corresponden al servidor ssh. Estos puertos solamente se utilizan durante la ejecución de lamboot y *del lado del servidor*, es decir en cada una de las computadoras donde no se ejecuta lamboot sino que el comando lamboot dispara, vía ssh, los procesos lamd encargados de toda las tareas posteriores de ejecución de programas y de *mantenimiento* de la máquina paralela LAM/MPI (con comandos como lamsrhink, lamhalt, etc.).

5. Conclusiones

Desde la perspectiva de la sostenibilidad técnica de cómputo intercluster con bibliotecas como LAM/MPI, la situación es bastante complicada. A menos que las políticas de seguridad (normalmente implementadas sobre los firewalls) se relajen mucho o aún se eliminen, sería virtualmente imposible utilizar una de estas bibliotecas para cómputo paralelo intercluster. Nótese, además, que se está dejando de lado la perspectiva de rendimiento, que será necesario resolver o al menos caracterizar una vez resueltos los problemas de seguridad. Volviendo a la perspectiva de seguridad, como mínimo, se debería permitir:

- El disparo de tareas remotas con ssh.
- La interconexión de procesos en cualquiera de las máquinas usando los protocolos UDP y TCP donde tanto *clientes* como *servidores* (desde la perspectiva de los protocolos, no de las aplicaciones paralelas de usuario) pueden estar en cualquiera de las computadoras.
- La utilización en principio no controlada/limitada de puertos no privilegiados para las conexiones entre *clientes* y *servidores*.

El único de los problemas ya ampliamente resueltos es el primero, el disparo de tareas remotas con ssh. Para los dos puntos restantes, sería necesaria la habilitación de todos los puertos no privilegiados para clientes y servidores de los protocolos TCP y UDP. Por un lado, esto implica abrir al exterior del cluster todos los puertos no privilegiados y por el otro, se debe asumir que todos los routers involucrados permiten el tráfico peer-to-peer, algo que todavía no está completamente garantizado. El tráfico peer-to-peer es considerado especialmente peligroso en ambientes académicos y más aún en el contexto de clusters abiertos al exterior, con múltiples computadoras expuestas y que en principio estarían dedicadas a cómputo paralelo.

Referencias

[1] Burns G., R. Daoud, J. Vaigl, "LAM: An Open Cluster Environment for MPI", Proceedings of Supercomputing Symposium, pp. 379-386, 1994. Available at http://www.lam-mpi.org/download/files/lam-papers.tar.gz

[2] Flynn M., "Very High Speed Computing Systems", Proc. IEEE, Vol. 54, 1966.

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

[4] Gropp W., 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,

[5] Gropp W., E. Lusk, N. Doss, A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard", Parallel Computing, Vol. 22, No. 6, pp. 789-828, Sep. 1996.

[6] Snir M., S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra., MPI: The Complete Reference, Volume 1 - The MPI-1 Core, 2nd edition. The MIT Press, 1998.

[7] Squyres J. M., A. Lumsdaine, "A Component Architecture for LAM/MPI", Proceedings, 10th European PVM/MPI Users'Group Meeting, pp. 379-387, 2003, Venice, Italy, Springer-Verlag Lecture Notes in Computer Science 2840, September/October 2003.

[8] Stevens R., TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley Longman, Inc., 1994.

[9] Tinetti F. G., Aróztegui W., "Instalación y Configuración de ssh para Cómputo Intercluster", Reporte Técnico PLA-002-2005, III-LIDI, Facultad de Informática, UNLP, CeTAD, Facultad de Ingeniería, UNLP, Argentina, Junio 2005. Disponible en https://lidi.info.unlp.edu.ar/~fernando/publis/intercl1.pdf

[10] LAM/MPI homepage, http://www.lam-mpi.org/