Mecánica Computacional Vol. XXIV
A. Larreteguy (Editor)
Buenos Aires, Argentina, Noviembre 2005

PERFORMANCE DE RED EN CLUSTERS BEOWULF

Carolina León Carri[*‡], **Carlos García Garino**[†] **y Guillermo Marshall**[*]

[*]Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires,
Ciudad Universitaria, Pabellón I, 1428 Capital Federal, Argentina
e-mail: mcarri@dc.uba.ar, e-mail: marshalg@mail.retina.ar
web page: http://www.lsc.dc.uba.ar

‡Instituto de Astronomía y Física del Espacio, FCEN, UBA.

†LAPIC, Instituto Tecnológico Universitario,
Universidad Nacional de Cuyo y CONICET,
Casilla de Correo 947, 5500 Mendoza, Argentina.
e-mail: cgarcia@itu.uncu.edu.ar

Palabras clave: cluster, HPC, performance, red.

Resumen.

En la última década la computación paralela tuvo un gran resurgimiento gracias a los clusters tipo Beowulf, sistema de cálculo paralelo de alta performance (High Performance Computing - HPC). Debido a que la programación paralela se basa en el pasaje de mensajes a través de la red es importante que el rendimiento de la misma sea óptimo. El objetivo de este trabajo es presentar un estudio de la performance de red y optimización de clusters Beowulf a través del análisis de la compleja relación existente entre la tecnología de red y la arquitectura del nodo. Se muestra que para obtener la máxima performance de Gigabit Ethernet es fundamental analizar cuidadosamente los elementos subyacentes. Se evalúa el throughput de la red a nivel de protocolo TCP sobre distintas arquitecturas de hardware mediante el benchmark NetPIPE. También se analiza el throughput de red de las funciones de envío y recepción de mensajes de MPICH. El trabajo realizado permite elaborar recomendaciones acerca del diseño de clusters tipo Beowulf con referencia a la arquitectura del nodo y la tecnología de red utilizada.

1. INTRODUCCIÓN

Los clusters de computadoras tipo Beowulf[1] surgieron en la NASA a fines de 1993 cuando el investigador Donald Becker pensó en desarrollar un recurso de supercómputo sin tener que utilizar una supercomputadora. El objetivo principal fue resolver problemas complejos en forma más económica.

Un cluster tipo Beowulf es un sistema de cálculo paralelo basado generalmente en el sistema operativo Linux, que consiste exclusivamente en componentes denominados *commodity* (obtenibles en el comercio) y redes de interconexión standard de modo tal que los nodos (CPU's) y los elementos de red que componen el equipo se obtienen con facilidad en el mercado. La red está dedicada para el uso privado del cluster. Debido a que la programación paralela se basa en el pasaje de mensajes a través de la red es importante que el rendimiento de la misma sea óptimo. Poseen software funcional y hardware que permiten que el cluster sea visto y administrado como un sistema único.

El objetivo de este trabajo es presentar un estudio de la performance de red y optimización de clusters Beowulf a través del análisis de la compleja relación existente entre la tecnología de red y la arquitectura del nodo. Se muestra que para obtener la máxima performance de *Gigabit Ethernet* es fundamental analizar cuidadosamente los elementos subyacentes.[2] El máximo *throughput* alcanzable depende fuertemente del bus PCI (*Peripheral Component Interconnect*) del nodo,[3] de la velocidad del procesador,[4] de las características de la placa de red y del *tuning* de los parámetros TCP (tamaño de ventanas de congestión y tamaño de bloques)[5] como así también de los *drivers* utilizados. Se evalúa el *throughput* de la red a nivel de protocolo TCP utilizando distintas arquitecturas de hardware (AMD® Athlon, Intel® Pentium4, Intel® Xeon y AMD® Opteron 64). Para esto se utiliza el módulo TCP del 'benchmark NetPIPE'.[6] Con los resultados obtenidos se realizan los gráficos de *throughput* según el tamaño de bloque transferido, de saturación de la red y de firma ethernet. Éstos últimos permiten analizar el máximo tamaño de paquete para el cual se logra un incremento en el *throughput* y la latencia de la red respectivamente. Por otro lado, se realizan mediciones para distintos tamaños de MTU (*Maximum Transfer Unit*) de 3000 y 9000 bytes. También se analiza el *throughput* de red de las funciones de envío y recepción de mensajes de MPICH.[7] El trabajo realizado permite elaborar recomendaciones acerca del diseño de clusters tipo Beowulf con referencia a la arquitectura del nodo y la tecnología de red utilizada.

El trabajo se encuentra organizado de la siguiente manera, en la segunda sección se presenta la teoría necesaria para explicar los experimentos. En la tercera sección se describe el benchmark NetPIPE, que se utilizó para realizar pruebas sobre redes *Fast Ethernet* y *Gigabit Ethernet*, en ambos casos sobre el cluster Speedy. Luego se comunican los resultados obtenidos sobre diferentes clusters de producción (HOPE, Mercurio y Reloaded), en todos los casos empleando una red *Gigabit Ethernet* y, finalmente, las conclusiones del trabajo. El hardware empleado se describe en el apéndice 1.

2. TECNOLOGÍA DEL CLUSTER

En este apartado se presentan los elementos que conforman un cluster tipo Beowulf: Arquitectura del nodo; Tecnología de red del cluster y los elementos *lógicos* del cluster: el protocolo de red, TCP-IP en este caso y la biblioteca de pasajes MPI.

2.1. Arquitectura del nodo

A grandes rasgos, el camino seguido en una transferencia de datos a través de una red implica tres cargas en memoria. Del lado del receptor, el adaptador de red accede directamente a memoria (*Direct Memory Access - DMA*) para cargar los datos en el buffer receptor, el CPU lee los datos de dicho buffer y los escribe al de la aplicación. En una transmisión se realiza el paso inverso. Las cargas en memoria dependerán de la aplicación que esté siendo utilizada, tamaños de buffers y el comportamiento de la cache.

El tráfico total de memoria, lecturas/escrituras, es a través del *Front Side Bus - FSB*. La Figura 1 muestra el tráfico de memoria durante la transmisión de un mensaje. La Figura 2 muestra el tráfico de memoria durante la recepción de un mensaje.

Envío de mensajes.

1. El CPU lee el buffer de la aplicación
2. El CPU escribe los datos al buffer del socket
3. La placa de red (NIC) accede directamente a memoria y lee el buffer del socket

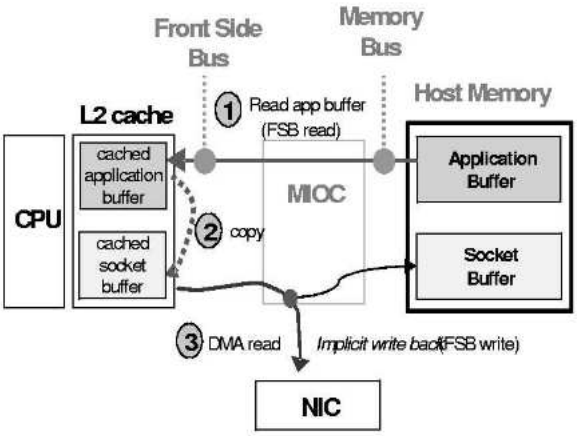

Figura 1: Envío de mensajes.[4]

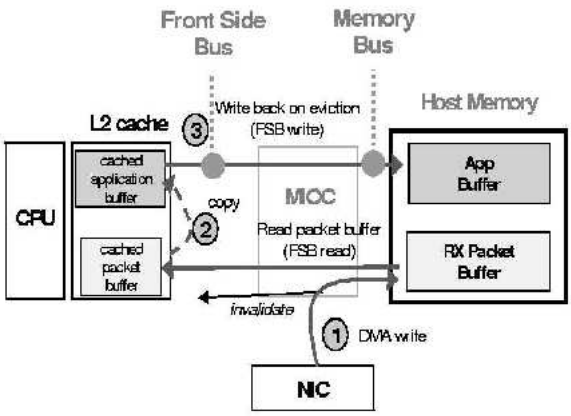

Figura 2: Recepción de mensajes.[4]

Recepción de mensajes.

1. La placa de red (NIC) accede directamente a memoria y escribe el buffer del socket
2. El CPU lee el buffer del socket

3. El CPU escribe los datos al buffer de la aplicación

El *bus PCI (Peripheral Component Interconnect)* se ha convertido en el standard que se emplea actualmente en las comunicaciones internas de los PC's. El *bus* ha ido evolucionando en el tiempo, en función del incremento de velocidad y prestaciones que muestran los diferentes componentes del equipo: microprocesador, FSB, memoria RAM, etc. Actualmente se tiende a utilizar el *bus* PCI–X, aunque todavía no es sencillo conseguir hardware para este tipo de *buses*. El Cuadro 1 muestra el ancho de banda, frecuencia y tasa de transferencia de las diferentes versiones de *bus* PCI.

El *bus* PCI es compartido entre varios dispositivos, por su naturaleza paralela. Aquellos dispositivos que se encuentren conectados al *bus* harán uso del mismo rotando cíclicamente. No solo comparten el *bus* sino que la frecuencia quedará definida en base al dispositivo de menor frecuencia, disminuyendo así la performance de los dispositivos más rápidos.[3] Por ejemplo, el *bus* PCI está compartido por el CPU, la memoria, la placa de video, las placas de red, etc.

La gran mayoría de los equipos existentes en general poseen placas de red *Fast Ethernet*, ya sea instaladas en una ranura PCI o bien *on board*. Actualmente se dispone de *motherboards* que incluyen placas de red *Gigabit Ethernet* o bien se pueden comprar en el mercado este tipo de placas e instalarlas en ranuras PCI. Mientras que la tasa de transferencia del PCI convencional es más que suficiente para redes *Fast Ethernet* los 32 bits/33 Mhz convencionales ya no brindan las prestaciones adecuadas para *Gigabit Ethernet*, como se observa en el Cuadro 1, ya que se debería disponer de manera dedicada el *bus* para uso de la placa de red.

La migración de un cluster de *Fast Ethernet* a *Gigabit Ethernet* no es trivial, porque muchas veces la arquitectura del nodo no permite obtener las tasas de transferencia nominales de *Gigabit Ethernet* y la relación costo/beneficio del cambio no es razonable.

Arquitectura	Ancho del bus	Frecuencia del bus	Ancho de banda del bus (Bytes)	Ancho de banda del bus (bits)
PCI 1.x 2.x	32–bits	33 Mhz	133 MBps	1 Gbps
PCI 2.2	32–bits	66 Mhz	264 MBps	2 Gbps
PCI 2.2/PCI–X	64–bits	66 Mhz	533 MBps	4.2 Gbps
PCI–X	64–bits	100 Mhz	800 MBps	6.4 Gbps
PCI–X	64–bits	133 Mhz	1 GBps	8 Gbps

Cuadro 1: Características del throughput nominal del Bus PCI/PCI-X.[3]

2.2. Tecnología de redes

La eficiencia de un cluster de alta performance depende en gran parte de la tecnología de red utilizada para interconectar sus nodos. Esto se debe a que los programas paralelos utilizan intercambio de mensajes entre los nodos involucrados en la resolución de un problema.

En la actualidad se emplean redes Ethernet conmutadas eliminando de esta manera el acceso al medio por contienda, basado en el clásico CSMA/CD.[8] En el caso de los clusters Beowulf es

deseable conectar solo un nodo por boca del *switch*, de esta manera la transmisión entre cada nodo y el *switch* se realiza en modo *full duplex*, materializando la denominada *microsegmentación*.

Tanto para las redes *Fast Ethernet* como *Gigabit Ethernet* es importante introducir los conceptos de latencia y ancho de banda, que permiten caracterizar el rendimiento de la red. El **ancho de banda** indica la cantidad de datos que pueden ser transferidos a través de un enlace por unidad de tiempo. La **latencia** es el tiempo necesario para transferir un dato desde un origen al destino, generalmente se refiere a las demoras debidas al procesamiento del dato de red. El tiempo de ida y vuelta se denomina **Round Trip Time (RTT)**.

El ancho de banda y la latencia están relacionados. Mientras que el máximo ancho de banda teórico es fijo (dado por el hardware de red utilizado), el ancho de banda real se ve afectado por la latencia de la red, es decir por el tiempo mínimo necesario para transferir un paquete de dato desde un punto a otro. Mucha latencia en un corto período de tiempo puede causar cuellos de botella que no permiten llenar el enlace, por lo tanto decrece el ancho de banda. También se ve afectado por el *overhead* del hardware y del sistema operativo. El ancho de banda práctico lo llamamos **throughput**. La percepción de la velocidad de la red es una síntesis de éstas dos unidades de medida.

Otro elemento importante que caracteriza a Ethernet es el tamaño del *frame* o **MTU (Maximum Transfer Unit)**, fijado en 1518 bytes. Algunas placas de red *Gigabit Ethernet* y *switches* de altas prestaciones permiten configurar el MTU para alcanzar valores de hasta 9000 bytes logrando así incrementar el *throughput*. Se debe tener en cuenta que si el MTU es muy pequeño, la máquina estará respondiendo seguido, mientras que si es muy grande puede haber errores y los paquetes tendrían que ser retransmitidos.

Ethernet contempla la posibilidad de agrupar dos o más enlaces en forma transparente para el usuario, característica conocida como **Link Aggregation**. Para materializar ésta posibilidad se debe disponer de un sistema operativo, placas de red y *switches* capaces de administrarla. Linux posee un driver de red denominado *bonding* que permite implementar esta configuración, que constituye una buena opción para lograr mayor performance de red a bajo costo. En el contexto de *bonding* se puede definir la política de balance de carga entre las dos placas, por ejemplo Round-Robin que envía en forma circular un paquete a cada placa disponible. Esta forma de trabajo necesita contar con conmutadores con mayor cantidad de bocas.

2.3. Protocolo de red TCP-IP

El protocolo estándar TCP-IP define al datagrama **IP (Internet Protocol)** como la unidad de información enviada a través de la red y provee las bases para la distribución de paquetes. Si bien el tamaño máximo de un datagrama es 64 Kbytes, en el clusters Beowulf prima el hardware de red, discutido en la sección anterior.

TCP (Transmission Control Protocol) es un protocolo *confiable* pues garantiza que los datos llegarán a destino y *orientado a conexión* pues simula un túnel entre el emisor y el receptor. Básicamente una conexión TCP consta de las siguientes partes:

emisor con un buffer para el envío de mensajes cuyo tamaño está dado en bytes

receptor con un buffer para la recepción de mensajes cuyo tamaño está dado en bytes y puede ser diferente al anterior

Dichos buffers, también llamados *ventana de congestión*, cambian de tamaño dinámicamente en el transcurso de una conexión TCP controlando así el flujo de datos.

El protocolo TCP provee *confiabilidad* debido al uso de paquetes de *acknowledge* (ACK), estos paquetes son enviados por el receptor para confirmar que se ha recibido un paquete determinado. Si el emisor no recibe el ACK en un determinado período de tiempo, reenvía el paquete. Si recibe el ACK continúa enviando paquetes. El tamaño de la ventana de congestión del emisor indicará cuantos paquetes puede enviar el mismo mientras que espera el ACK de cada uno de ellos. El tamaño de la ventana del receptor estará condicionado por la rapidez que el mismo tenga para enviar el ACK del paquete recibido y liberar espacio en el buffer.

La performance depende del tamaño de las ventanas y de la velocidad a la cual la red transmita los paquetes. Cuando se incrementa el tamaño de la ventana es posible reducir por completo los momentos ociosos de la red. Una buena configuración del tamaño de la ventana deslizante, mantiene a la red completamente saturada de paquetes, obteniendo un mejor *throughput*. Luego, el máximo *throughput* está condicionado por el tamaño de ambos buffers. La siguiente fórmula permite calcular el máximo ancho de banda dado el tamaño del buffer emisor y el tiempo de respuesta del enlace.

$$\text{ancho de banda} = \frac{\text{tamaño del buffer}}{\text{RTT}}$$

Por la misma fórmula, sabiendo el ancho de banda teórico de la red y el RTT podemos calcular el tamaño óptimo de la ventana de congestión para lograr un buen *throughput*.

2.4. Biblioteca de pasaje de mensajes MPI

El paradigma de *pasaje de mensajes* Message Passing Interface - MPI,[9] es utilizado usualmente en distintas clases de computadoras paralelas, especialmente en aquellas que poseen memoria distribuida. Es un estándar de funciones para llevar a cabo el pasaje de mensajes en la programación paralela. Puede invocarse desde C y Fortran. Una función de pasaje de mensajes se encarga simplemente de transmitir datos de un proceso a otro a través de la red. En este trabjo se emplea MPICH.[9] Entre otras funciones provee los siguientes tipos de operaciones descriptas en detalle en:[10]

- Combinación de contexto y grupos de mensajes a través de un *comunicador*
- Comunicación punto a punto
  - estructuras de buffers y tipos de datos
  - modos de comunicación: normal (bloqueante y no bloqueante), sincrónica, buffered
- Comunicación Colectiva

- gran variedad de rutinas para pasar datos entre procesos
- definición de grupos

- Control de errores

En los sistemas UNIX, MPICH utiliza la biblioteca llamada *p4*[11,12] como interface sobre la capa TCP. Las transferencias de datos sólo se llevan a cabo durante las llamadas a las funciones de la biblioteca MPICH. Se puede optimizar el uso de la biblioteca *p4* mediante la configuración de diversas variables en el entorno del usuario. Por ejemplo, la variable *P4_SOCKBUFSIZE* se utiliza para definir el tamaño del buffer para el socket, que se define en bytes. Se recomienda actualizar el valor por defecto que tiene la variable *P4_SOCKBUFSIZE* de 32 KB por 128 KB, ya que de esta manera se aumenta el *throughput* de MPI.[13]

Por otro lado, se puede optimizar el uso del protocolo TCP utilizando la opción *-p4sctrl* al ejecutar un trabajo por línea de comandos. Dicha opción permite definir distintas características del socket TCP, pero sólo se recomienda modificar el *bufsize*, en este caso se define en KB.[7,13] Por ejemplo, si se quiere definir el buffer del socket en 64 KB (por defecto es de 16 KB), se debe ejecutar:

```
# mpirun -np N mpptest -p4sctrl bufsize=64
```

dónde N indica el número de procesadores a utilizar.

Es importante mencionar que la biblioteca *p4* utiliza tres protocolos de pasaje de mensaje dependiendo del tamaño del paquete a transmitir.[14] Para paquetes menores o iguales a 16 KB se emplea ***Short***, en este caso los datos se envían junto con la información de control. ***Eager*** se utiliza para enviar paquetes cuyo tamaño esta comprendido entre 16 y 128 KB, y los datos se envían sin esperar la solicitud del receptor. Finalmente para paquetes cuyo tamaño es mayor a 128 KB, se emplea ***Rendezvous*** y en este caso se demora el envío de los datos hasta que el receptor los pida. Esta estrategia en el tratamiento de los paquetes se realiza con el fin de optimizar la latencia en el envío de mensajes cortos y alcanzar mayor ancho de banda para los largos.

3. EXPERIMENTOS REALIZADOS SOBRE REDES FAST ETHERNET Y GIGABIT ETHERNET

En esta sección se analiza el rendimiento de los protocolos TCP y la biblioteca de pasaje de mensajes MPICH, para lo cual se utiliza el benchmark NetPIPE, descripto en la subsección 3.1.

Las experiencias que se comunican, se realizaron con el cluster Speedy, cuyas características se presentan en el Cuadro 2 del Apéndice 1 de este trabajo, y son una reseña del trabajo de Licenciatura de la primer autora,[2] en cuyo trabajo se basa la metodología empleada.

En la subsección 3.2 se discuten los experimentos realizados con una red *Fast Ethernet*. También se estudió la posibilidad de agrupar dos o más canales de red, facilidad conocida como *Link Aggregation*. El estudio se repitió para el caso de una red *Gigabit Ethernet*, ver subsección 3.3 para lo cual también se utilizó el cluster Speedy.

3.1. Descripción del benchmark NetPIPE

NetPIPE, Network Protocol Independent Performance Evaluator [6,7,15] como su nombre lo indica, es una herramienta desarrollada para medir la performance de comunicación entre dos máquinas independientemente del protocolo utilizado. Posee diferentes módulos según la capa de red que se quiere medir TCP, MPI, MPI2, entre otros.

NetPIPE envía un mensaje que rebota entre dos nodos repetidas veces, y de esta manera se obtiene el tiempo de transmisión para cada tamaño de mensaje. El tamaño de los mensajes se elige a intervalos regulares y se aplican pequeñas perturbaciones para una evaluación completa de las capas de comunicación tanto a nivel de hardware como de software. Así se obtiene la latencia de pequeños mensajes, tomada como la mitad del *round trip time* (RTT) para mensajes de hasta 8 bytes y un gráfico del *throughput* de un amplio rango de tamaño de mensajes. El módulo MPICH permite evaluar el throughput utilizando pasaje de mensajes sincrónico y asincrónico.

Los resultados pueden ser guardados en archivos de texto que contienen tamaño de paquete, *throughput* y tiempo de transferencia para cada caso. Luego se pueden realizar gráficos de interés como:[15]

Throughput Ethernet Se grafica el *throughput* en función del tamaño de bloque transferido.

Firma Ethernet Se grafica el *throughput* versus el tiempo de transferencia en escala logarítmica obteniendo así un gráfico de aceleración de la red. La latencia de la red coincide con el primer punto de tiempo.

Saturación Ethernet Se grafica el tamaño de bloque versus el tiempo ambos en escala logarítmica. Se define como *punto de saturación* aquel a partir del cual al incrementar el tamaño del bloque el tiempo crece linealmente. El intervalo de tiempo luego del punto de saturación se determina *intervalo de saturación*.

3.2. Fast Ethernet y Link Aggregation

Con el benchmark NetPIPE a nivel de TCP y MPICH se realizaron experiencias para *Fast Ethernet* y *Link Aggregation* con el fin de medir el *throughput*, la latencia y los puntos de saturación en el cluster Speedy.

En las Figuras 3, 4 y 5 muestran los resultados obtenidos con el benchmark NetPIPE a nivel de protocolo TCP y MPICH, tanto para para *Fast Ethernet* como para *Link Aggregation* (driver *bonding* en Linux) en el cluster Speedy. Las ventanas de congestión TCP se definieron en 16 KB para el buffer de envío y 85 KB para el buffer de recepción. Para MPICH el tamaño de los buffers de las funciones de envío y recepción de mensajes se configuró en 32 KB, valor que posee por defecto la variable P4_SOCKBUFSIZE.

La Figura 3 muestra el gráfico de *throughput* (Megabits por segundo - Mbps) en función del tamaño de bloque transferido (bits). Se observa que *Fast Ethernet* para TCP presenta un crecimiento armónico y constante alcanzando un 90 % del ancho de banda disponible en el medio de conexión. MPICH se comporta del mismo modo perdiendo apenas un promedio del

7 % de performance en comparación con TCP. El mejor *throughput* de MPICH se alcanza para paquetes de tamaño 2, 4, 6 y 8 KB donde la diferencia es solo de un 1.5 %. La pérdida de performance con respecto a TCP es previsible pues al agregar una capa de abstracción sobre el protocolo TCP se genera un *overhead* propio de la implementación de las funciones intermedias utilizadas para el pasaje de mensajes.

Para realizar las pruebas con *Link Aggregation* y el driver *bonding*, se instalaron dos placas por nodo las cuales se configuraron en modo balance de carga.[16] El benchmark NetPIPE se ejecutó en condiciones similares al caso de *Fast Ethernet*. Para la capa TCP se obtuvo una performance del 80 % de la capacidad del medio, 200 Mbps en este caso, para paquetes mayores a 32 KB. Sin embargo se observan irregularidades para algunos tamaños de paquetes: 6 KB, 16 KB, 24 KB, 64 KB y 256 KB. Para el caso de MPICH se alcanza una performance aproximada de 130 Mbps para paquetes de tamaño mayor a 64 KB, que significa una mejora del 38 % respecto a los resultados de MPICH obtenidos para *Fast Ethernet*. Luego *bonding* es una buena opción para obtener mayor performance a bajo costo siempre y cuando el *switch* soporte *Link*

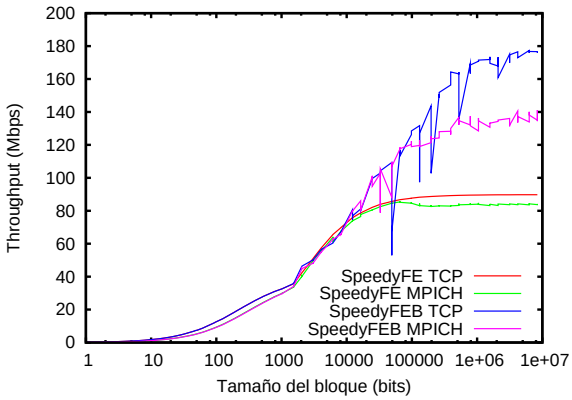

Figura 3: Throughput de Speedy. FE: Fast Ethernet, FEB: FE Bonding.

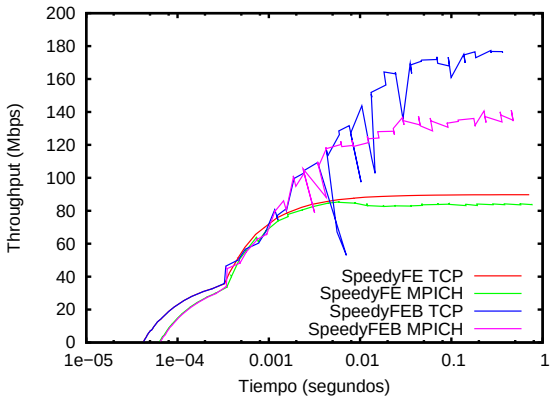

Figura 4: Gráfico de Firma Ethernet de Speedy. FE: Fast Ethernet, FEB: FE Bonding.

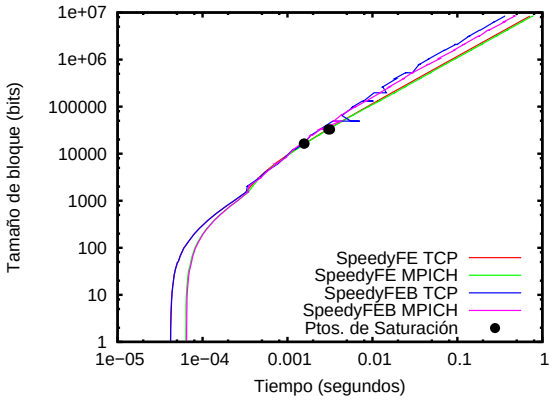

Figura 5: Gráfico de Saturación Ethernet de Speedy. FE: Fast Ethernet, FEB: FE Bonding.

Configuración	Latencia μs	Punto de Saturación (KB)
TCP	42	2
MPICH	65	4

Figura 6: Latencia y Punto de Saturación.

Aggregation y posea el número de bocas necesarias para asignar dos canales por cada nodo del cluster.

En la Figura 4, gráfico de Firma Ethernet, se representa el *throughput* en función del tiempo de transferencia, este último en escala logarítmica. Este gráfico representa dos características importantes de una red TCP, la latencia y la aceleración. Como se observa en las curvas de la Figura 4 se obtiene la latencia de TCP igual a 42 μs, coincide tanto para *Fast Ethernet* (una sola placa de red) como para *Link Aggregation* (dos placas de red configuradas en *bonding*). Para MPICH se obtiene una latencia del orden de 65 μs, mayor que la anterior debido a que los datos se procesan a más alto nivel.

En el gráfico de Saturación Ethernet, Figura 5, se observa que el punto de saturación para *Fast Ethernet* se obtiene para paquetes de tamaño 2 KB mientras que para *Link Aggregation* el tamaño es de 4 KB. En este caso los puntos de saturación de MPICH coinciden con los de TCP. En el *intervalo de saturación* el gráfico crece a velocidad constante, y a partir de allí, el *throughput* de la red (TCP/MPICH) no puede mejorarse considerablemente incrementando el tamaño del bloque.

3.3. Gigabit Ethernet

Para avanzar en el estudio de la tecnología de red, se evalúa el benchmark NetPIPE utilizando un redes *Gigabit Ethernet*. Primero se evalúa el *throughput* de red del cluster *Speedy* a nivel de protocolo TCP y MPICH. En la subsección 3.3.1 se analiza la influencia del tamaño de las ventanas de congestión de TCP. En la subsección 3.3.2 se analiza la influencia del tamaño de los *frames* Ethernet en el rendimiento de la red, para lo cual se investigan además del MTU convencional (1518 bytes), *jumbo frames* de tamaño 3000 y 9000 bytes, respectivamente. En todos los casos el *throughput* se presenta en función del tamaño de bloque transferido y se realizan los gráficos de firma y de saturación ethernet.

3.3.1. Estudio del tamaño de las ventanas TCP y MPICH

La influencia del tamaño de las ventanas TCP se observa en la Figura 7, en la cual se muestra el *throughput* en función del tamaño de los buffers de envío y recepción de paquetes del socket TCP. Se han procesado distintos casos partiendo de la configuración por defecto que corresponde a tamaños de 16 y 85 KB para los buffers de envío y recepción respectivamente. También se ha ensayado el tamaño del buffer de evío en 64 KB, activado con el parámetro *default2*. Finalmente se ha medido la respuesta para buffers de envío y recepción ambos del mismo tamaño, configurados para 64, 128, 256 y 512 KB respectivamente.

En la Figura 7 se observa que se pueden clasificar las curvas en distintos casos según su comportamiento:

1. buffers de 256 y 512 KB
2. buffers default y 128 KB
3. buffer de 64 KB

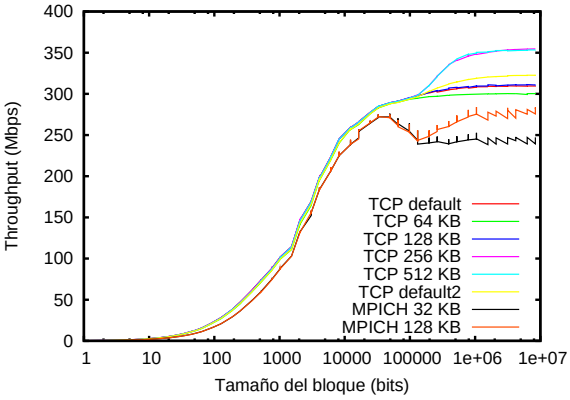

Figura 7: Throughput de Speedy con placas Gigabit Ethernet para distintos tamaños de buffer TCP. default: buffer de envío 16 KB, default2: buffer de envío 64 KB, ambos casos recepción 85 KB. MPICH ambas pruebas se realizaron con el buffer TCP default modificando el buffer de MPICH (P4).

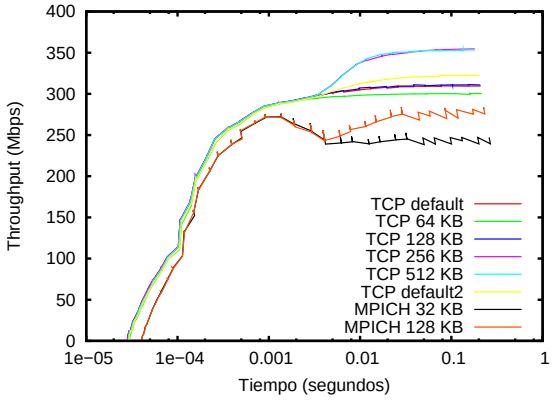

Figura 8: Gráfico de Firma Ethernet de Speedy con placas Gigabit Ethernet para distintos tamaños de buffer TCP. default: buffer de envío 16 KB, default2: buffer de envío 64 KB, ambos casos recepción 85 KB. MPICH ambas pruebas se realizaron con el buffer TCP default modificando el buffer de MPICH (P4).

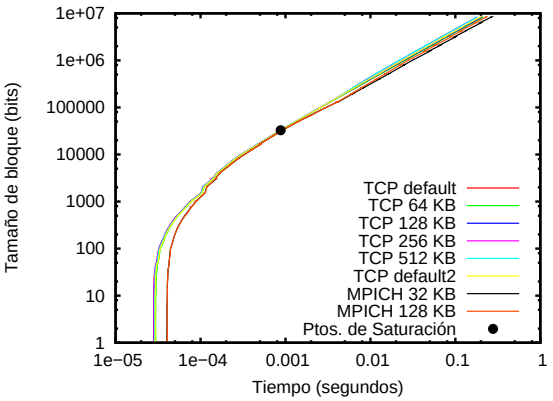

Figura 9: Gráfico de Saturación Ethernet de Speedy con placas Gigabit Ethernet para distintos tamaños de buffer TCP. default: buffer de envío 16 KB, default2: buffer de envío 64 KB, ambos casos recepción 85 KB. MPICH ambas pruebas se realizaron con el buffer TCP default modificando el buffer de MPICH (P4).

Configuración	Latencia μs	Punto de Saturación (KB)
TCP	28	4
MPICH	40	4

Figura 10: Latencia y Punto de Saturación.

4. buffer default2
5. MPICH

El *throughput* de MPICH se muestra para buffers P4 de 32 KB y 128 KB, se evaluaron las operaciones de envío y recepción de mensajes sincrónicas. Ambos casos fueron ejecutados sobre buffers TCP configurados en 16 KB para buffer de envío y 85 KB para buffer de recepción

de mensajes.

Una conclusión relevante que se desprende de la Figura 7 es que el máximo *throughput* alcanzado es del orden de 350 Mbps, lejos de la velocidad nominal de *Gigabit Ethernet*. Se desprende del resultado obtenido que no es sencillo incrementar el rendimineto de la red actualizando solamente el hardware de la misma (placas y *switch*).

La respuesta obtenida para los casos 1 a 5 es prácticamente la misma para paquetes cuyo tamaño es menor a 24 KB. En el caso 1, buffers de 256 KB y 512 KB se logra un incrementeo del 13 % en comparación con la configuración default del sistema y del buffer de 128 KB. Para el caso 3, buffer de 64 KB disminuye el throughput en un 3 %. Finalmente para la configuración de buffers default2 se logra una pequeña mejora del 4 % en comparación con la configuración por defecto. A partir de los resultados obtenidos se recomienda configurar el buffer en 256 KB.

En las pruebas de MPICH con su configuración de buffers de envío y recepción de mensajes por defecto (32 KB), el throughput disminuye en promedio un 15 % con respecto a TCP. También se evaluó la performance de MPICH incrementando el buffer mediante la variable P4_SOCKBUFSIZE a 128 KB. Se logró en promedio una mejora del 10 % sobre el caso por defecto para paquetes mayores o iguales a 16 KB. Cabe destacar que el pico más alto se alcanza para paquetes mayores o iguales a 128 KB, valor que se debe al protocolo que utiliza la interface P4 para envío de mensajes, *Rendezvous* en este caso. En función de los resultados obtenidos, se aconseja configurar la variable P4_SOCKBUFSIZE en 128 KB para todos los usuarios del sistema.

El gráfico de Firma Ethernet, que se muestra en la Figura 8, permite observar que la latencia es igual a 28 μs en todos los casos para TCP y alcanza un valor de 40 μs para MPICH. La latencia de un mensaje de MPICH en *Gigabit Ethernet* es tan grande como la latencia TCP en *Fast Ethernet* y *Link Aggregation*.

Finalmente, en el gráfico de Saturación de la Red, Figura 9, se observa que los puntos de saturación de MPICH coinciden con los de TCP siendo de 4 KB. Aquí nuevamente en el *intervalo de saturación* el gráfico crece a velocidad constante, por lo tanto a partir de allí, el *throughput* no puede mejorarse considerablemente incrementando el tamaño del bloque.

3.3.2. Estudio de la influencia del MTU

Otra posibilidad para mejorar el *throughput* es variar el tamaño del MTU, ya que al utilizar *jumbo frames* se requieren menor número de paquetes para enviar o recibir la misma cantidad de información y, consecuentemente, disminuyen los cálculos a nivel de TCP y el *overhead* resultante, de esta manera se obtiene un *throughput* mayor.

En la gran mayoría de los casos Ethernet fija el MTU en 1518 bytes, sin embargo algunos equipos de interconexión, en general de alto costo y rendimiento, admiten los llamados *jumbo frames*. En el trabajo se analiza la influencia del tamaño de MTU utilizando el valor usual de 1518 bytes, 3000 y 9000 bytes. En los tres casos los buffers TCP fueron definidos en 64 KB para el buffer de envío y 85 KB para el buffer de recepción.

Para realizar las pruebas se conectaron dos de los nodos espalda contra espalda mediante un

cable cruzado pues el *switch* utilizado, no soportaba MTU mayor a 1518 bytes. Las placas de red empleadas soportan esta opción que se configura mediante el comando de linux *ifconfig*. Por ejemplo para 3000 bytes se ejecutó:

```
# ifconfig eth0 mtu 3000
```

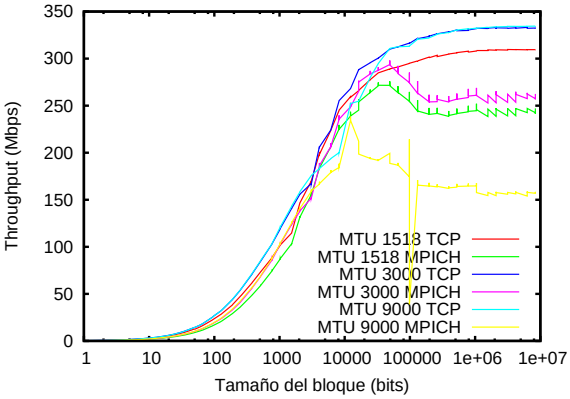

Figura 11: Throughput de Speedy con placas Gigabit Ethernet para MTU igual a 1518, 3000 y 9000.

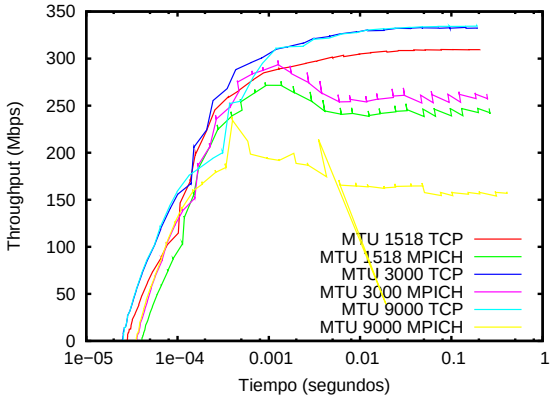

Figura 12: Gráfico de Firma Ethernet de Speedy con placas Gigabit Ethernet para MTU igual a 1518, 3000 y 9000.

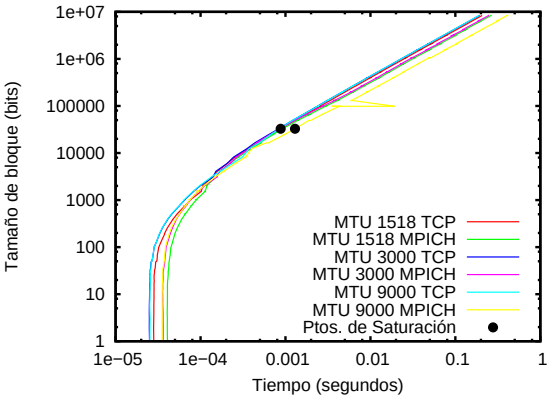

Figura 13: Gráfico de Saturación Ethernet de Speedy con placas Gigabit Ethernet para MTU igual a 1518, 3000 y 9000.

Configuración	Latencia μs	Punto de Saturación (KB)
TCP 1518	28	4
MPICH 1518	40	4
TCP 3000	25	4
MPICH 3000	36	4
TCP 9000	25	4
MPICH 9000	35	4

Figura 14: Latencia y Punto de Saturación.

Cuando se fija el MTU en 3000 bytes se logra una mejora de 20 Mbps respecto del MTU convencional. También se logra una mejora similar con un MTU de 9000 bytes, pero su comportamiento es un poco inestable y por debajo del caso de 3000 bytes. Es posible que en el caso de el 9000 bytes, el hardware del nodo no sea lo suficientemente rápido utilizar la capacidad de ancho de banda que provee *Gigabit Ethernet*, debido a los cálculos que requiere TCP-IP.

Para el tamaño de MTU por defecto, 1518 bytes, los ensayos realizados con dos nodos conectados espalda contra obtuvieron el mismo resultado que el estudio realizado con los nodos conectados a través del *switch Gigabit Ethernet*.

En la Figura 11 se compara el *throughput* obtenido para los tres casos estudiados. Prácticamente no se observan diferencias para paquetes de tamaño menor a 8 bytes, ya que en general para paquetes tan pequeños el benchmark envía un sólo paquete TCP y devuelve en promedio la latencia obtenida, con lo que estos casos no son útiles para analizar el throughput. También se observa que para cada caso la función crece abruptamente en cierto intervalo de tiempo, es allí donde se obtiene un throughput sustancialmente mayor al incrementar el tamaño del bloque. Luego de dichos intervalos, no se logra mayor beneficio al incrementar el tamaño del paquete. Esta última observación se puede ver gráficamente realizando un gráfico de saturación de la red.

La latencia de la red se determina mediante el gráfico de Firma Ethernet que se muestra en la Figura 12. Para MTU igual a 1518 bytes en TCP, se obtiene una latencia de 28 μs, que se mejora para los MTU de 3000 y 9000, para los cuales se obiene una latencia de 25 μs en ambos casos. Para MPICH nuevamente se incrementa la latencia siendo de 40 μs para MTU igual a 1518 bytes, 36 μs para 3000 bytes y 35 μs para 9000 bytes.

El gráfico de Saturación Ethernet, que se muestra en la Figura 13, permite determinar que el punto de saturación para *Gigabit Ethernet* en todos los casos resulta igual 4 KB, con prescindencia del tamaños de MTU utilizado.

4. ESTUDIOS EN CLUSTERS DE PRODUCCIÓN

Con el fin de profundizar el estudio realizado se emplea el benchmark NetPIPE para medir el rendimiento de la red en distintos clusters de producción: HOPE, Reloaded y Mercurio. Los resultados obtenidos se comparan con los de Speedy, utilizado como equipo de referencia.

En primer lugar se analiza el comportamiento de los clusters HOPE, Reloaded y Speedy, para lo cual se utilizan placas de red *Gigabit Ethernet* instaladas en una ranura PCI de los distintos equipos. Para el estudio se utilizó la placa 3Com® modelo 3c2000.[17] Los resultados obtenidos se discuten en la sección 4.1. Por otra parte se realizaron mediciones en los clusters HOPE, Mercurio y Reloaded, los cuales poseen la característica de tener la placa de red *on board*. Estos resultados se discuten en la subsección 4.2. Finalmente, en la subsección 4.3 se muestra la importancia de disponer los drivers de red actualizados para que el sistema operativo administre el hardware de manera adecuada.

4.1. Resultados para placas insertadas en ranura PCI

Con la citada placa de red 3Com® 3c2000,[17] utilizada en los ensayos del cluster Speedy en la sección anterior, se realizaron pruebas sobre los clusters Reloaded, basado en Intel® Pentium4 y HOPE, AMD® Opteron 64 bits, para lo cual las placas de red se colocaron sobre una ranura del bus PCI. En todos los casos se utilizó la configuración por defecto para los buffers TCP, 16 KB para el buffer de envío y 85 KB para el buffer receptor.

Como se observa en la Figura 15 se obtiene un *throughput* del orden de 600 Mbps para

Reloaded, alrededor de 400 Mbps para HOPE y 350 Mbps para Speedy, con lo cual puede observarse que el rendimiento de la red *Gigabit Ethernet* depende en gran medida del hardware del nodo. También se observan diferencias en la latencia medida, como se muestra en el gráfico de la Figura 16. Finalmente, la Figura 17 muestra los puntos de saturación.

Resulta difícil elaborar una explicación concluyente para justificar la diferencia de rendimiento obtenido. En los tres casos el bus PCI posee una tasa de transferencia del orden de 1000 Mbps (32 bits/33 Mhz). La arquitectura que mejor funcionó, corresponde a un nodo Intel® Pentium 4 del cluster Reloaded que posee un FSB de 800 Mhz y caché L2 de 1 MB. De las pruebas realizadas surge que en el caso de redes *Gigabit Ethernet* se debe estudiar cuidadosamente la placa de red a utilizar, el cluster debe ser diseñado para garantizar un rendimiento equilibrado de la red y del nodo.

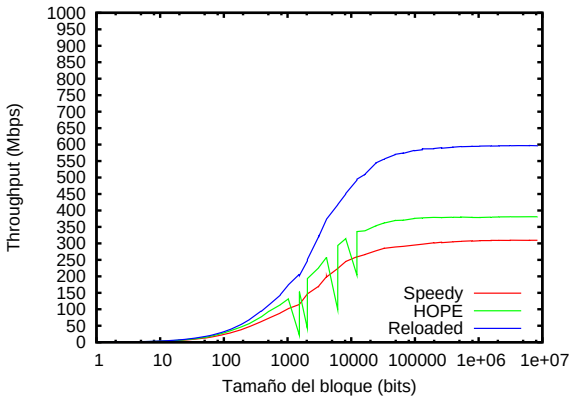

Figura 15: Comparación del Throughput de la placa Gigabit Ethernet 3Com 3c2000 en distintas arquitecturas.

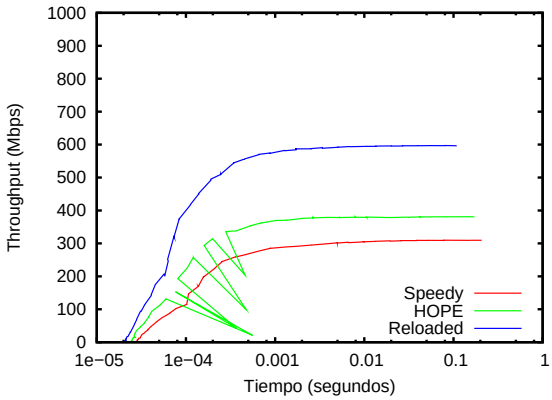

Figura 16: Gráfico de Firma Ethernet de la placa Gigabit Ethernet 3Com 3c2000 en distintas arquitecturas.

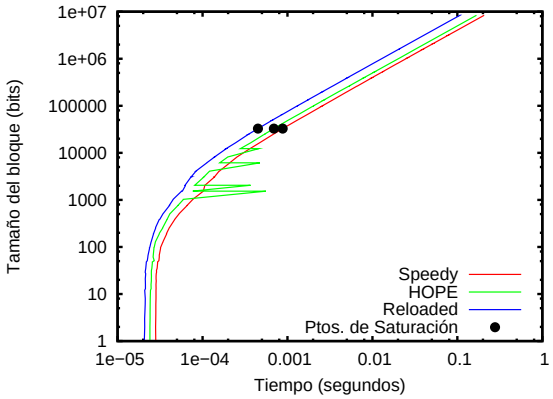

Figura 17: Gráfico de Saturación Ethernet de la placa Gigabit Ethernet 3Com 3c2000 en distintas arquitecturas.

Configuración	Latencia μs	Punto de Saturación (KB)
Speedy	28	4
HOPE	24	4
Reloaded	20	4

Figura 18: Latencia y Punto de Saturación.

4.2. Comparación entre clusters con placa de red on board

Para complementar los estudio del rendimiento de los clusters empleados en este trabajo, se discuten y comparan los resultados obtenidos con los clusters Mercurio, Reloaded y HOPE, ya que los tres poseen placa de red Intel® *Gigabit Ethernet* de características similares con la particularidad de estar incluídas en el *motherboard*.

Mercurio y HOPE pueden considerarse equipos de alta gama, orientados a servidores, como se desprende del Cuadro 2 del Apéndice 1, en el cual se compara el hardware, mientras que el hardware de Reloaded puede considerarse dentro de del contexto de una estación de trabajo de alto rendimiento. En este sentido cabe decir que si bien los tres equipos se pueden obtener en el mercado quizás no son representativos del hardware convencional.

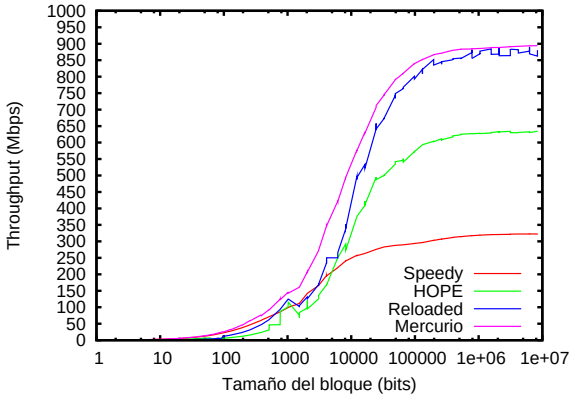

Figura 19: Comparación del Throughput en distintas arquitecturas.

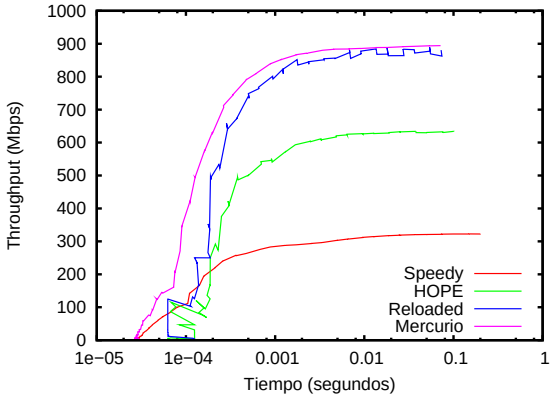

Figura 20: Gráfico de Firma Ethernet en distintas arquitecturas.

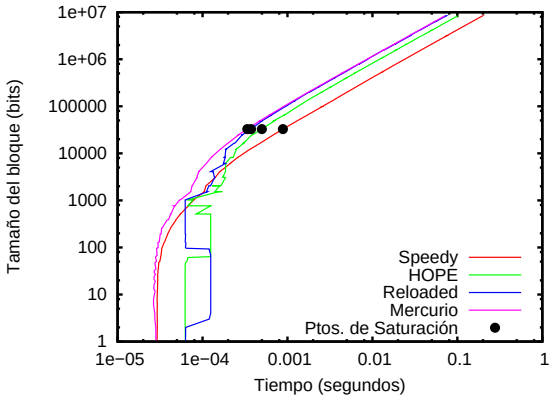

Figura 21: Gráfico de Saturación Ethernet en distintas arquitecturas.

Configuración	Latencia μs	Punto de Saturación (KB)
Speedy	28	4
HOPE	62	4
Reloaded	63	4
Mercurio	28	4

Figura 22: Latencia y Punto de Saturación.

Mercurio posee características avanzadas: 2 procesadores Intel® Xeon 2.8 Ghz, 2 MB de caché L2, FSB 800 Mhz, 4 GB de memoria RAM, bus PCI-X y placas de red *on board Gigabit Ethernet* para Fibra Óptica. El *throughput* alcanzado para este cluster, el mayor de todos los

obtenidos, fue del orden de 900 Mbps. Reloaded posee un microprocesador Intel® Pentium 4 HT, caché L2 1 MB, FSB 800 Mhz, 1 GB de RAM y placa de red *on board* con un canal de comunicaciones dedicado que garantiza del orden de 2000 Mbps de ancho de banda. En este caso se obtuvo un rendimiento de 890 Mbps con el benchmark NetPipe. Finalmente, HOPE posee dos microprocesadores AMD® Opteron de 64 bits, la placa de red está instalada *on board*, pero emplea para las comunicaciones el bus PCI convencional. Parece coherente que en este caso solo se alcancen alrededor de 600 Mbps con las pruebas de NetPipe.

Las Figuras 19, 20 y 21, muestran gráficos que comparan el *throughput*, el gráfico de Firma Ethernet y el gráfico de Saturación de la red para las diferentes arquitecturas que poseen placa de red Intel® instalada *on board*. Se ha añadido a manera de comparación los resultados obtenidos con el cluster Speedy que posee una tecnología de hardware de nodo de menores prestaciones que los anteriores y que originalmente fue diseñado para una red *Fast Ethernet*.

De las pruebas realizadas surge que parece importante contar con equipos que muestren equilibrio entre las prestaciones del microprocesador, bus FSB y cachés incluidos, el bus de comunicaciones de la placa de red y la tecnología de red utilizada. En este sentido el hardware de HOPE, que posiblemente posee el microprocesador de mayores prestaciones, muestra un rendimiento de la red del órden del 60 % del nominal, frente a los otros dos casos, Reloaded y Mercurio, en los cuales se obtiene un rendimiento muy bueno, del órden del 90 % del nominal. En este sentido, como ya se ha señalado, no parece una buena práctica instalar placas de red *Gigabit Ethernet* en ranuras PCI, aunque la reciente aparición del bus PCI-X seguramente solucionará este inconveniente.

Es importante destacar que se observan resultados dispersos para la latencia, los cuales deben revisarse con mayor profundidad, ya que en este caso influyen el hardware del nodo, la placa de red y el conmutador que materializa la interconexión de la red *Gigabit Ethernet*.

4.3. Influencia del driver de la placa de red

Es importante actualizar los drivers de la placa de red en todos los equipos. Los mismos vienen incluídos dentro de la distribución del sistema operativo Linux, y no siempre se incluye dentro de las mismas, la última versión disponible del cluster. Las Figuras 23, 24 y 25, muestran el rendimiento obtenido para la misma placa de red Intel® utilizando diferentes drivers de red. Como se observa en los mismos la diferencia de rendimiento obtenida aconseja muy fuertemente no descuidar este efecto, ya que en el caso del cluster Reloaded prácticamente se duplica el rendimiento al actualizar el driver.

5. CONCLUSIONES

A partir de las experiencias realizadas con el benchmark NetPIPE, utilizando los diferentes clusters disponibles, tanto con redes *Fast Ethernet* y *Gigabit Ethernet* se han obtenido resultados que permiten establecer algunos comentarios y recomendaciones de interés para el diseño de clusters de alta performance.

El bus PCI posee una tasa de transferencia suficiente para redes *Fast Ethernet*. En este caso

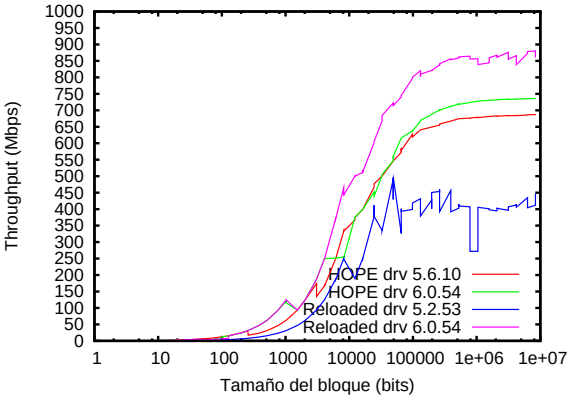

Figura 23: Throughput de HOPE y Reloaded para distintos drivers de red Intel® Pro 1000. Buffers TCP de envío 16 KB, recepción 85 KB.

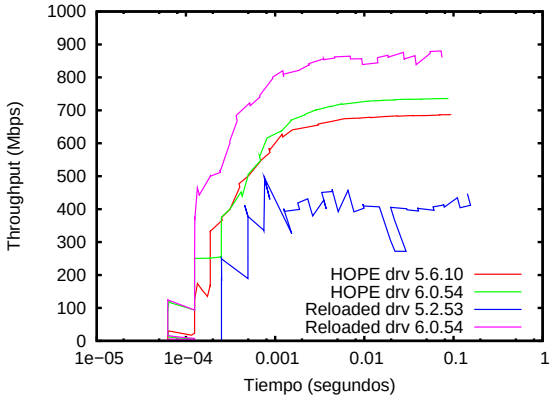

Figura 24: Gráfico de Firma Ethernet HOPE y Reloaded para distintos drivers de red Intel® Pro 1000. Buffers TCP de envío 16 KB, recepción 85 KB.

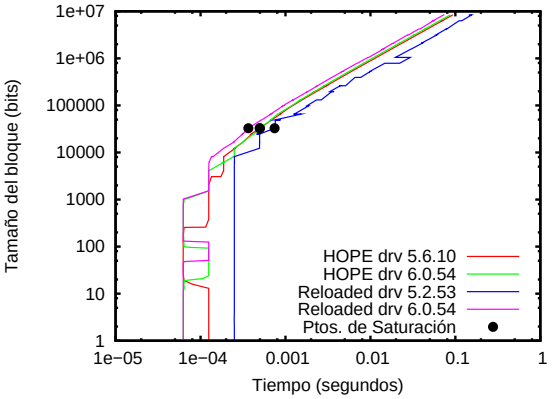

Figura 25: Gráfico de Saturación Ethernet de HOPE y Reloaded para distintos drivers de red Intel® Pro 1000. Buffers TCP de envío 16 KB, recepción 85 KB.

Configuración	Latencia μs	Punto de Saturación (KB)
HOPE drv 5.7.6	124	4
HOPE drv 6.0.54	62	4
Reloaded drv 5.2.53	249	4
Reloaded drv 6.0.54	63	4

Figura 26: Latencia y Punto de Saturación.

el empleo de *Link Aggregation*, a través de *bonding*, incrementa el rendimiento en alrededor de 80 % para TCP y 40 % para MPICH. Los valores de configuración por defecto para TCP y MPICH resultan adecuados en este caso.

El diseño de un cluster con tecnología de red *Gigabit Ethernet* no es simple. Parece necesario utilizar buses PCI con una tasa de transferencia de al menos 2 Gbps (como mínimo 32 bits/66 Mhz) para garantizar el funcionamiento correcto de la placa de red, o bien disponer de un canal dedicado para la misma. En el caso de placas de red incluidas en el *motherboard* a veces se verifica esta prestación, pero no siempre es así. En general no se puede garantizar que se obtengan velocidades cercanas a las nominales con placas *Gigabit Ethernet* instaladas en ranuras PCI. En todos los casos es recomendable instalar drivers de red actualizados y realizar pruebas de rendimiento.

La migración de un cluster existente a *Gigabit Ethernet* es compleja. No siempre se pueden obtener anchos de banda cercanos a los nominales, ya que la arquitectura del nodo en general no esta acorde con las prestaciones de la red y conviene analizar cuidadosamente el beneficio que se obtiene frente a la inversión a realizar. En este caso es necesario tener en cuenta la tasa de transferencia del bus PCI y del bus FSB, que pueden constituirse en los cuellos de botella del nodo y por ende del cluster.

Mientras que los valores por defecto de TCP y MPICH son adecuados para *Fast Ethernet*, parece importante analizar los valores de configuración de las variables P4 para el caso de *Gigabit Ethernet*. En este sentido es importante optimizar las aplicaciones de gran escala.

6. AGRADECIMIENTOS

Los autores agradecen la ayuda financiera recibida de la Agencia Nacional de Promoción Científica y Tecnológica, ANPCyT, a través del proyecto PICTR 184 del FONCyT.

7. APÉNDICE: HARDWARE UTILIZADO

Cluster	CPU	FSB	Cache KB	RAM MB	Motherboard	Bus PCI	Placa de red	Switch
Speedy	AMD® Athlon 1200 MHz	200	256	512 DDR 333 MHz	PCchip® M810	32-bit/66Mhz PCI 2.2 264 MBps	3Com® 3c905cx	3Com® Office Connect
Speedy	AMD® Athlon 1200 MHz	200	256	512 DDR 333 MHz	PCchip® M810	32-bit/66Mhz PCI 2.2 264 MBps	3Com® 3c2000	TRENDnet® TEG-S240TX
Reloaded	Intel® Pentium 4 3 Ghz	800	1024	1024 DDR 400	Intel® D875PBZ	32-bit/66Mhz PCI 2.2 264 MBps	Intel® 82547EI	TRENDnet® TEG-240WS
HOPE	AMD® Opteron 1.8 Ghz	-	1024	2048 DDR SDRAM	Tyan® Tiger K8W (S2875)	32-bit/33Mhz PCI-1.2 133 MBps	Intel® 82541EI	LinkSys® SR2024
Mercurio	Intel® Xeon 2.8 Ghz	800	2048	4096 DDR SDRAM	Intel®	64-bit/66Mhz PCI-X 533 MBps	Intel® 82544EI	AVAYA® P880

Cuadro 2: Características del hardware utilizado.

Speedy: Mandrake 9.2, kernel 2.4.22-21mdk
Reloaded: Red Hat AS 3.0, kernel 2.4.21-20.ELsmp
Mercurio: Red Hat 7.3, 2.4.26
HOPE: Red Hat AS 3.0, kernel 2.4.29

REFERENCIAS

[1] *Clusters Beowulf.* http://www.beowulf.org.

[2] C. León Carri. Análisis de performance y optimización en clusters beowulf. Master's thesis, UBA, (Agosto 2004).
[3] Intel. Pci express ethernet networking. smoothly migrate networking to the next-generation, industry-standard serial i/o architecture. (2003).
[4] H. H. Hum J. P. Patwardhan A. P. Foong, T. R. Huff. Tcp performance re-visited.
[5] B. L. Tierney. Tcp tuning guide for distributed application on wide area networks. Technical report, Data Intensive Distributed Computing Group. Laurence Berkeley National Laboratory., (Febrero 2001).
[6] X. Chen y T. Benjegerdes D. Turner, A. Oline. Integrating new capabilities into netpipe. Technical report, Ames Laboratory - Iowa State University.
[7] D. Turner y X. Chen. Protocol dependent message passing performance on linux clusters. Technical report, Ames Laboratory - Iowa State University, (Septiembre 2002).
[8] D. E. Comer. *Internetworking with TCP/IP. Volume I, Principles, protocols and architecture*. (1995).
[9] *MPI Standard*. http://www.mcs.anl.gov/mpi/index.html.
[10] P. Pacheco. *Parallel Programming with MPI*. Universidad de San Francisco, (1997).
[11] *P4*. http://www.mcs.anl.gov/ lusk/p4/.
[12] T. Disz B. Glickfeld E. Lusk R. Overbeek J. Patterson J. Boyle, R. Butler and R. Stevens. *Portable Programs for Parallel Processors*. Holt, Rinehart and Winston, (1987).
[13] W. Gropp y E. Lusk. *Installation and User's Guide to MPICH, a Portable Implementation of MPI Version 1.2.5. The ch p4 device for Workstation Networks.*
[14] R.Ñamyst O. Aumage, G. Mercier. Mpich/madeleine: a true multi-protocol mpi for high performance networks. Technical Report LIP, ENS-LYON 46, All'ee d'Italie 69364 Lyon Cedex 07, France.
[15] Mikler y Gustafson Snell. Netpipe: A network protocol independent performance evaluator. Technical report, Ames Laboratory - Iowa State University.
[16] T. Davis. *Linux Ethernet Bonding Driver mini-howto*.
[17] 3Com, http://www.3com.com. *Manual de usuario. Tarjeta de interfaz de red Gigabit de 3Com 3C2000*, (Abril 2003).
[18] N. Snyder. Ibm @server - ibm linux clusters. Technical report, (Noviembre 2003).