Replicación de datos en PVFS2 para conseguir tolerancia a fallos

Erik Nieto Tovar[1], Raúl Hernández Palacios[1], Hugo E. Camacho Cruz[1], Antonio F. Díaz García[1], Mancia Anguita López[1], Julio Ortega Lopera[1]

***Resumen*— El uso de los discos de los nodos de un cluster como sistema de almacenamiento global es una solución barata; pero para que sea una solución viable se debe afrontar el problema que supone la frecuencia de fallos en discos y en los nodos del cluster. Estos fallos provocan errores en el acceso a ficheros de las aplicaciones. El número de fallos en el acceso a ficheros es especialmente importante en plataformas con un elevado número de nodos y con sistemas de ficheros paralelos, como PVFS. En este trabajo se muestra cómo se ha añadido a la segunda versión del sistema de ficheros paralelo PVFS replicación de datos para conseguir tolerancia a fallos y la repercusión en las prestaciones de la implementación.**

Palabras clave—**Paralelismo, replicación, sistemas de ficheros paralelos.**

I. INTRODUCCIÓN

Actualmente, numerosas aplicaciones científicas trabajan con grandes cantidades de datos que necesitan cargarse o almacenarse en disco; por ejemplo, bioinformática, predicción del tiempo, modelado del clima, procesamiento de imágenes de satélites, etc. Para mejorar prestaciones en el acceso a disco en los computadores de gama media y alta se utilizan sistemas de E/S basados en SAN (Storage Area Network, Fig. 1 (a)). Pero, teniendo en cuenta la extensión en el uso de cluster y con el fin de mejorar la relación prestaciones/coste se están utilizando también sistemas de ficheros que permiten aprovechar el almacenamiento de los nodos del cluster (Fig. 1 (b)). De esta forma se evita el coste del sistema SAN. Entre estos sistemas de ficheros que permiten a las aplicaciones ver como sistema de almacenamiento el conjunto formado por los discos de todos los nodos del cluster, están Lustre [1] y PVFS2 (www.pvfs.org). Ambos son sistemas de ficheros paralelos; es decir, permiten que varios nodos puedan acceder en paralelo al mismo fichero y que un nodo pueda acceder a múltiples trozos de un fichero concurrentemente. Lo consiguen distribuyendo un fichero entre diferentes discos.

Teniendo en cuenta la probabilidad de fallo de un disco ([2, 3]) será viable el uso de los discos de los nodos de un cluster como sistema de almacenamiento global si se provee un mecanismo para proporcionar tolerancia a fallos. Esto es especialmente importante si el número de nodos es elevado o si el sistema de ficheros es paralelo. Por ejemplo, en [3] se ha encontrado un AFR (Annual Fail Rate) de 8% para discos con dos años de antigüedad. Esto supone que si en un sistema hay 600 discos podrían fallar, como media, cuatro discos al mes. También influye en la frecuencia de fallo en los accesos en sistemas de ficheros distribuidos los fallos del servidor de datos (memoria, procesador, etc.). Además, la probabilidad de que falle el acceso a un fichero en sistemas de ficheros paralelos es mayor que la probabilidad de que falle un disco o un servidor porque un fichero está distribuido entre diferentes discos y servidores.

PVFS es un sistema de ficheros paralelo gratuito para Linux que actualmente se encuentra en su segunda versión. Para implementar alta disponibilidad en PVFS2 la documentación sugiere utilizar Heartbeat (www.linux-ha.org). Pero esta alternativa requiere que el almacenamiento esté compartido (con una SAN). Para permitir tolerancia a fallos, evitando compartir el almacenamiento, se propone aquí añadir a PVFS2 replicación de datos en dos servidores. De esta forma se evita el coste de usar una SAN y se pueden aprovechar los discos incluidos en los nodos del cluster. En la implementación actual un bloque de datos se encuentra almacenado en dos servidores distintos.

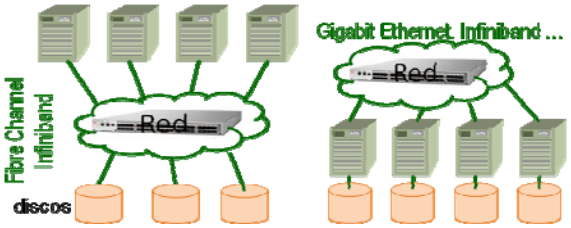
En la primera versión de PVFS se han implementado dos alternativas para conseguir tolerancia a fallos ([4, 5]). En [4] se implementa un esquema híbrido que utiliza RAID 5 para escrituras grandes y RAID 1 para pequeñas escrituras. Esta implementación debe incluir un mecanismo que adapte dinámicamente el modo de redundancia para evitar las prestaciones más reducidas de RAID 5 con respecto a RAID 1 para escrituras pequeñas. Para RAID 5 se necesita además serializar las escrituras que necesitan modificar el mismo bloque de paridad. En [5] se implementa un esquema RAID 10; es decir, los bloques distribuidos con RAID 0 de PVFS se replican en servidores utilizados como espejos (RAID 1). Para evitar que los servidores espejo estén inactivos cuando se lee, la implementación lee la mitad de un bloque del servidor principal y la otra mitad del servidor espejo.

En este artículo presentaremos una replicación propuesta para PVFS2 y analizaremos la sobrecarga que introduce en la escritura y en la lectura (de la copia) en caso de fallo del servidor principal.

El resto del artículo se ha organizado de la siguiente forma: la Sección II describe la arquitectura de PVFS2;

[1] Dpto. de Arquitectura y tecnologia de computadores, Univ. Granada email:{enieto,rpalacios,hcamacho, afdiaz ,manguita,julio}@atc.ugr.es

en la Sección III se presenta el método utilizado para realizar la replicación; la Sección IV muestra los resultados obtenidos y finalmente en la Sección V se presentan algunas conclusiones.



(a) (b)

Fig. 1. (a) Configuración SAN. (b) Aprovechamiento de los discos de los nodos del cluster como sistema de ficheros virtual.

II. Arquitectura del sistema de ficheros PVFS2

El sistema de ficheros paralelo PVFS2 ha sido desarrollado a partir de la primera versión de éste, mejorando la modularidad, la flexibilidad entre módulos y añadiendo una fuerte integración con MPI-IO.

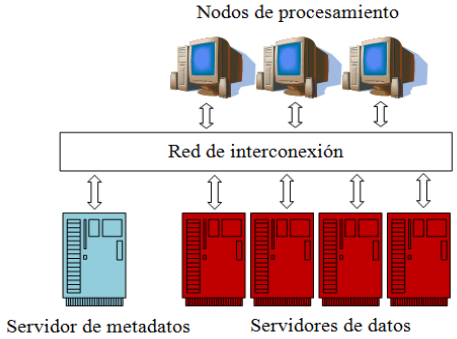
A. ESTRUCTURA HARDWARE

PVFS2 está compuesto principalmente por tres elementos (Figura 2):

1. Servidores de metadatos (metaservers). Los servidores de metadatos son los encargados de almacenar toda la información de metadatos de los ficheros dentro del espacio de nombres del sistema. Entre otra información, almacenan los atributos, distribución, número y valor de datahandles. Cuando los clientes desean realizar una operación en un fichero determinado, primero deben comunicarse con este servidor para obtener toda la información necesaria para llevar a cabo dicha operación. Los metadatos se almacenan principalmente en dos bases de datos: una para almacenar los metadatos en sí, y otra para almacenar los datahandles. Estos últimos son los nombres de los ficheros donde se almacenarán los datos en los servidores de datos.

2. Servidores de datos (dataservers). Los servidores de datos (también denominados de E/S) son los encargados de almacenar una parte de los ficheros de datos del sistema. Los datos se almacenan en archivos Linux (bstreams). Estos servidores almacenan las rutas de acceso a dichos ficheros en bases de datos. En conjunto, los servidores de datos, proporcionan a PVFS2 el subsistema de almacenamiento. Al calcular el espacio disponible en el sistema de ficheros, se realiza una suma interna del espacio disponible en cada servidor. El cálculo es muy básico, simplemente realiza una recopilación del espacio disponible en todos los servidores, y multiplica el menor de ellos por el número de servidores disponibles.

3. Nodos de procesamiento o clientes. Estos nodos son los encargados de acceder al sistema de ficheros en sí. Para poder hacerlo disponen de una librería llamada pvfs2lib. A su vez, también existe un módulo, insertado en el kernel de Linux, que ofrece un punto de acceso al sistema a través del sistema virtual de ficheros (VFS) de Linux. Adicionalmente se puede acceder al sistema a través de la librería de MPI-IO.



15. Fig. 2 Arquitectura de PVFS2

B. ARQUITECTURA SOFTWARE

Los módulos que conforman la capa software de PVFS2 y su interacción se muestran en la Figura 3. PVFS2 utiliza una semántica no bloqueante entre las distintas capas. El cliente se comunica con la interfaz del sistema PVFS2 a través de la capa de aplicación utilizando diferentes tipos de interfaces a nivel de usuario (Kernel-VFS, MPI-IO, etc.)

La capa de interfaz de sistema es una API puesta a disposición de los clientes, que utiliza una serie de máquinas de estado para ejecutar las operaciones solicitadas por la capa superior. Estas máquinas de estado lanzan nuevas operaciones, ejecutadas y controladas por la capa JOB, cada una de las cuales tiene un número asociado para ser identificadas.

Las operaciones ejecutadas por la capa JOB, se envían a los servidores a través de la capa FLOW, que se encarga de mover tanto operaciones como datos de los clientes a los servidores.

La capa FLOW, a su vez, utiliza la capa BMI (Buffered Message Interface), para poder tener accesos a la red. BMI pone a disposición de la capa FLOW, distintos módulos que dan soporte a diversos tipos de red. Estos módulos hacen transparente al cliente el tipo que se está utilizando. Actualmente BMI soporta TCP (por defecto) [6], Infiniband, Gamma, Via y Myrinet.

Las capas de software en el servidor son prácticamente las mismas que en los clientes, excepto que los servidores tienen una capa llamada TROVE, que es la encargada de almacenar los datos en los dispositivos de almacenamiento. Esta capa actúa tanto en los servidores de metadatos como en los servidores de datos. En los primeros, dando acceso a las distintas bases de datos donde se almacenan los metadatos. En los segundos, dando acceso a las bases de datos que se utilizan para ubicar a un determinado datafile y también

brindando acceso a los archivos tipo LINUX que almacenan los trozos de archivo del sistema de ficheros.

En la Figura 3, se muestran tres partes importantes de la capa de software de PVFS2, que es diferente a la implementada en la primera versión. Por una parte la acache y la ncache, son dos caches que existen en el lado cliente, para almacenar de forma temporal los atributos de los ficheros que se han accedido últimamente, y los nombres de los ficheros, respectivamente. Y por otra parte las máquinas de estados finitos en el lado cliente y en el lado servidor. Estas máquinas de estado son las encargadas de ejecutar las distintas operaciones recibidas desde la capa de interfaz del sistema. Por ejemplo, existe una máquina de estados para la creación de ficheros en el lado cliente y una máquina para crear ficheros en el lado servidor. Estas máquinas de estado son un conjunto de pasos que van ejecutando las operaciones de acuerdo a los resultados de pasos anteriores y posteriores. Ha sido necesario el cambio de algunas de estas máquinas de estado para lograr la replicación de datos.

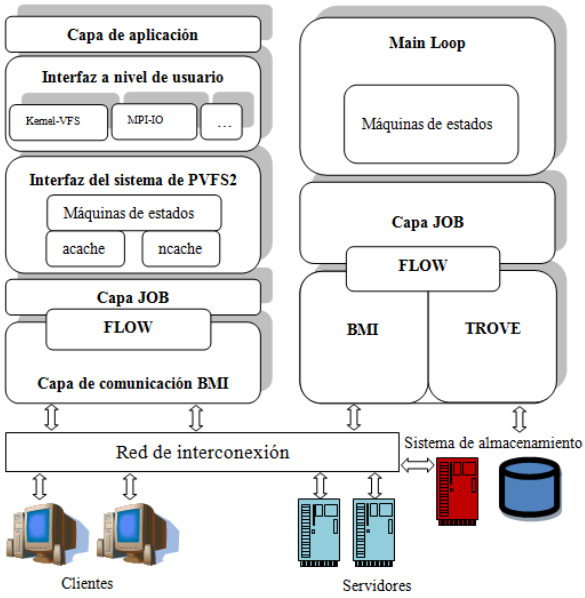


Fig. 3 Arquitectura software de PVFS2.

III. ESQUEMA DE REPLICACIÓN SECUENCIAL

A. *Distribución de datos en PVFS2*

La manera en que PVFS2 distribuye los datos a través de los servidores de E/S es almacenando bloques, por defecto de 64KB, en cada uno de ellos (RAID 0). Por defecto, reparte los bloques en Round-Robin. En la Figura 4 se muestra el proceso de asignación de bloques.

Cada servidor almacena un archivo tipo Linux con bloques del archivo original. Una de las principales problemáticas con este esquema es que, por ejemplo, para una operación de E/S sobre un fichero, los clientes deben interactuar con todos los servidores para poder reconstruir un trozo de fichero (en el caso de las lecturas), o para poder añadir datos en éste (en el caso de las escrituras). En la Figura 5 se muestra el proceso de reconstrucción de un fichero en una lectura, participando todos los servidores.

La desventaja de este procedimiento en los sistemas de ficheros paralelos es que al distribuir los datos de un archivo entre distintos dispositivos de almacenamiento, y posteriormente obtenerlos de cada uno de ellos (como se muestra en la Figura 5), en la presencia de un fallo de cualquiera de los servidores de E/S el archivo original quedará incompleto, y ya no se podrá hacer E/S sobre éste. Por tanto, la lectura de un trozo de un fichero se puede ver afectado por el fallo de alguno de los servidores en los que se encuentra distribuido el fichero. En la Figura 6 se puede ver gráficamente esta idea.

En este artículo proponemos un modelo de replicación secuencial, que consiste básicamente en tener ficheros de replicación en los servidores de E/S que servirán, en el caso de la caída de un servidor, para recuperar los datos del servidor contiguo.

En la siguiente sección se explica la forma en que fueron llevados a cabo los cambios a las máquinas de estados para conseguir la replicación.

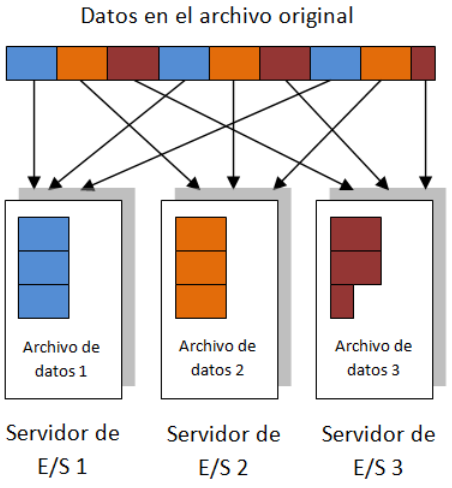


Fig. 4 Distribución de los bloques de datos en PVFS2.

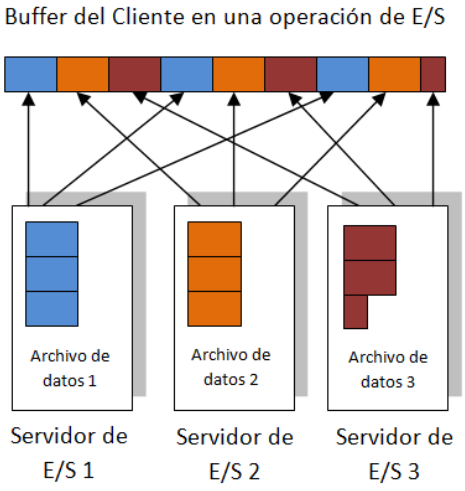


Fig.5 Reconstrucción de un archivo de PVFS desde los servidores de E/S.

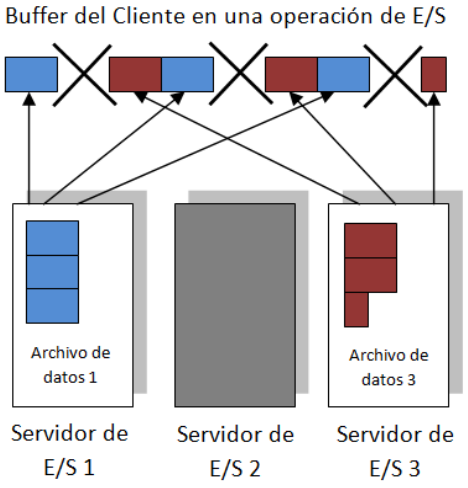


Fig. 6 Fallo de una servidor de E/S.

B. Proceso para la creación de un fichero en PVFS2 sin y con replicación.

La secuencia para la creación de un fichero en PVFS2 mediante la máquina de estados correspondiente es como se muestra a continuación:

Paso 1: Crear un metafile, en el servidor de metadatos para almacenar los atributos del fichero.
Paso2: Crear una colección de datafiles, para almacenar los datos en los distintos servidores de E/S (el número total dependerá de la distribución elegida por el llamador).
Paso3: Crear una entrada de directorio en el directorio padre, para agregar el fichero al espacio de nombres.
Paso4: Terminar la máquina de estados si es que todo ha ido correctamente.

Para realizar la replicación, fue necesario la modificación de estos pasos y la inclusión de uno nuevo, encargado de crear los nuevos datafiles, que se utilizarán en la E/S a ese fichero. La máquina de estados quedó de la siguiente manera:

Paso 1: Crear un metafile, en el servidor de metadatos para almacenar los atributos del fichero.
Paso2: Crear una colección de datafiles, para almacenar los datos en los distintos servidores de E/S (el número total dependerá de la distribución elegida por el llamador).
Paso3: Crear una colección de datafiles, para almacenar los datos de replicación en los distintos servidores de E/S (este número de datafiles, tendrá el mismo valor y la misma distribución que los datafiles anteriores).
Paso4: Crear una entrada de directorio en el directorio padre, para agregar el fichero al espacio de nombres.
Paso5: Terminar la máquina de estados si es que todo ha ido correctamente.

En la Figura 7 se muestra cómo se almacenan en los servidores el fichero que contendrá los datos originales y el que contiene las réplicas.

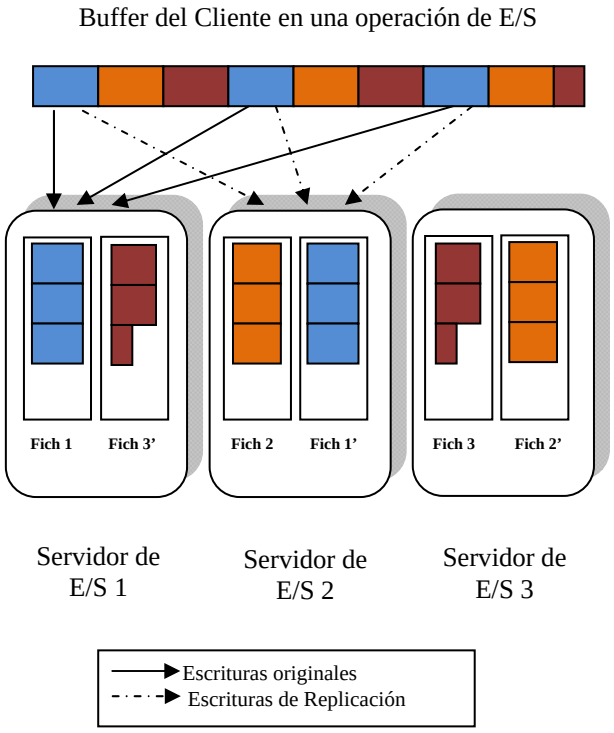


Fig. 7 Replicación de datos en los datafiles de replicación.

Cuando el cliente escribe en un fichero, escribirá inicialmente los datos en los ficheros originales, y posteriormente en los de replicación. Si se comienza a escribir los ficheros originales en el servidor i, los ficheros copia comienzan en el (i+1) mod n, donde n es el número de servidores. Todos los servidores, por tanto, contienen réplicas, al contrario que en las implementaciones en [4] y en [5] las que las réplicas estarán en la mitad de los servidores, en la otra mitad habrá sólo originales. Esto hace que para que en las lecturas intervengan todos los servidores y conseguir así mayores prestaciones, deben implementar un mecanismo adicional que permita leer un bloque mitad del servidor original y la otra mitad del servidor con la copia. En la implementación aquí propuesta no hay que implementar este mecanismo adicional para conseguir que se pueda leer concurrentemente de todos los servidores ya que todos ellos contienen originales y copias.

En la siguiente sección mostraremos la influencia de la replicación en las prestaciones de las E/S.

IV. Evaluación de Prestaciones

Las siguientes pruebas fueron realizadas en el cluster heterogéneo wlan, que cuenta con cinco nodos. Dos de estos nodos tienen un procesador Intel Core 2 Duo de 2.4GHz, 2 Gb de memoria RAM, Disco Duro de 500Gb. Además tres nodos con un procesador AMD Athlon 64 con 1Gb de memoria RAM y Disco Duro. Los nodos están conectados mediante un switch Gigabit Ethernet.

A. Experimentación con escrituras.

La experimentación se realizó con la siguiente configuración: 3, 4 y 5 servidores actuando como nodos

de E/S y un cliente accediendo a ficheros con tamaño ascendente desde 32KB hasta 1Gb. Al utilizar un único cliente las prestaciones están limitadas por las dos escrituras que tiene que realizar éste. PVFS2 usa en los experimentos bloques de 64 KB. La Figura 8 muestra el ancho de banda del sistema de ficheros en las escrituras. En esta se observa que el máximo ancho de banda de las escrituras se obtiene con archivos aproximados de 2MB. El máximo ancho de banda que se obtuvo fue de 80 MB/s aproximadamente, manteniéndose en 45MB/s a partir de un tamaño de archivo de 16MB/s.

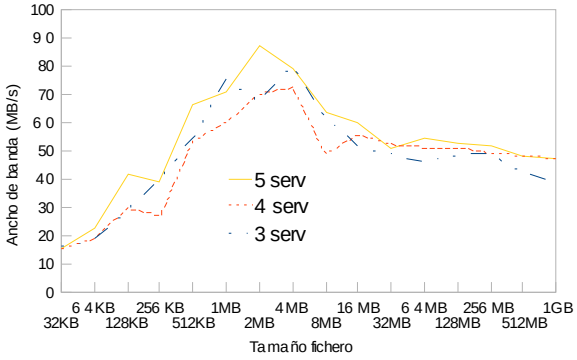


Fig. 8 Ancho de banda de escrituras sin replicación

En la Figura 9 el ancho de banda obtenido para las escrituras realizando la replicación de datos. En esta gráfica se puede observar que el máximo ancho de banda alcanzado es de 24 MB/s. El ancho de banda se mantiene, para las diferentes configuraciones, en un valor medio de 20MB/s.

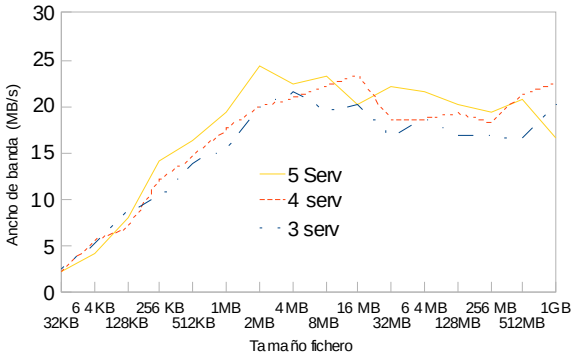


Fig. 9 Ancho de banda de escrituras con replicación.

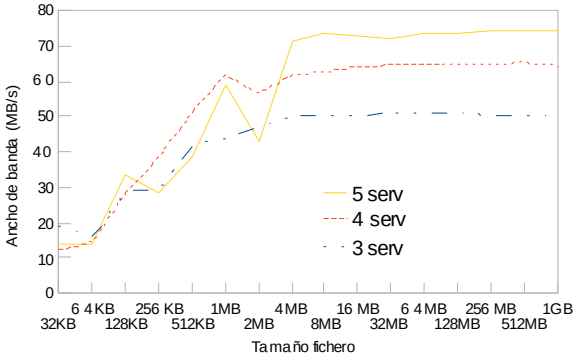


Fig. 10 Ancho de banda de las lecturas en PVFS

Según muestran los resultados, la escritura con replicación implica una penalización en el ancho de banda asintótico de aproximadamente el 60% con respecto a las escrituras en un sistema sin replicación. Se espera este nivel de penalización debido a las dos escrituras que tiene que realizar el cliente. En [5] se reportan penalizaciones del 50% con cuatro clientes.

B. Experimentación con lecturas

El test con las lecturas se realizó con las mismas configuraciones de las escrituras. En la Figura 10, se observa el ancho de banda de las lecturas en caso de no producirse ningún fallo. Se alcanzan los 70 MB/s para cinco servidores. En la Figura 11 se observa los anchos de banda obtenidos leyendo de las réplicas cuando un servidor ha caído. Sólo cae un servidor para las tres configuraciones, por eso la penalización en las prestaciones es mayor con la configuración de tres servidores que con cuatro o cinco.

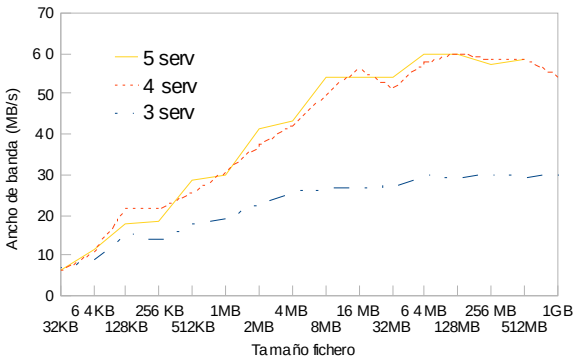


Fig. 11 Ancho de banda de PVFS leyendo de las replicas.

V. CONCLUSIONES Y TRABAJO FUTURO

Los resultados del test muestran que, como era de esperar, las escrituras disminuyen el ancho de banda en más de la mitad y que la penalización en las lecturas cuando falla un servidor depende del número de servidores en total a los que hay que acceder. Como trabajo futuro, se pretende mejorar las prestaciones aprovechando en mayor medida el hardware disponible (múltiples cores, múltiples interfaces de red) y usando otras alternativas de implementación de la réplica que tengan en cuenta la implementación particular de PVFS2.

AGRADECIMIENTOS

Este trabajo ha sido financiado mediante el proyecto P06-TIC-01935.

REFERENCIAS

[1] P. J. Braam, "The Lustre Storage Architecture," November. 2002.

[2] B. Schroeder and G. A. Gibson, "Understanding disk failure rates: What does an MTTF of 1,000,000 hours mean to you?" *Trans.Storage,* vol. 3, pp. 8, 2007.

[3] E. Pinheiro, W. Weber and L. A. Barroso, "Failure trends in a large disk drive population," in *FAST '07: Proceedings of the 5th USENIX Conference on File and Storage Technologies,* 2007, pp. 2-2.

[4] M. Pillai and M. Lauria, "A high performance redundancy scheme for cluster file systems," in *Proceedings of the 2003 International Conference on Cluster Computing,* 2003, pp. 216-223.

[5] Y. Zhu and H. Jiang, "CEFT: A cost-effective, fault-tolerant parallel virtual file system," *Journal of Parallel and Distributed Computing,* vol. 66, pp. 291-306, 2006.

[6] J. M. Kunkel and T. Ludwig, "Performance evaluation of the PVFS2 architecture," in 2007, pp. 509-516.