Evaluación de rendimiento del sistema de ficheros paralelo Expand Ad-Hoc en MareNostrum 4

Diego Camarmas-Alonso, Félix García-Carballeira, Alejandro Calderón-Mateos y Jesús Carretero[1]

Resumen— **Durante los últimos años las aplicaciones utilizadas en el campo de la ciencia están evolucionando hacia el análisis masivo de datos a través de *workflows* debido al crecimiento de áreas como la Inteligencia Artificial y el *big data*.**

Sin embargo, el mayor cuello de botella cuando se ejecutan este tipo de aplicaciones se encuentra en las operaciones de E/S. Para tratar de solventar este problema se está desarrollando el sistema de ficheros paralelo Expand Ad-Hoc. Permite crear particiones virtuales ad-hoc para incrementar el rendimiento de E/S en entornos de supercomputación. El objetivo de este trabajo es presentar una evaluación de este sistema de ficheros sobre el supercomputador MareNostrum 4. En la evaluación de Expand Ad-Hoc llevada a cabo en MareNostrum 4, se ha podido comprobar que su rendimiento y escalabilidad es globalmente superior al del sistema de ficheros paralelo GPFS.

Palabras clave— **Sistema de ficheros paralelo y distribuido, Expand Ad-Hoc, MareNostrum 4, GPFS.**

I. Introducción

En los últimos años las aplicaciones, principalmente en el campo de la ciencia, y muy especialmente en las crecientes áreas de Inteligencia Artificial o *big data*, han evolucionado hacia el tratamiento masivo de datos a través de flujos de trabajo, también conocidos como *wokflows*. Cada vez son más aplicaciones en estas crecientes áreas, como por ejemplo aplicaciones que permiten diagnosticar enfermedades mediante el análisis de imágenes de gran resolución, previsión meteorológica, simulación de turbulencias o dinámicas moleculares, entre otras.

El grupo de investigación ARCOS está desarrollando el sistema de ficheros Expand Ad-Hoc [2], que es un sistema de ficheros distribuido y paralelo que ha sido diseñado y optimizado para adaptarse a estas nuevas necesidades.

En [1] se presentó un prototipo inicial y una evaluación de rendimiento preliminar de Expand Ad-Hoc. En este nuevo trabajo se va a presentar la versión estable de este sistema de ficheros y una evaluación de rendimiento más completa.

En la versión estable de Expand Ad-Hoc se añade la interceptación de nuevas llamadas al sistema que no habían sido interceptadas en el prototipo inicial. Además, se añaden diversas optimizaciones que ayudan a incrementar el ancho de banda de E/S y se simplifica del despliegue de los servidores ad-hoc para mejorar la experiencia de usuario.

[1]Departamento de Informática, Universidad Carlos III de Madrid (UC3M), e-mail: {dcamarma,fgcarbal,acaldero,jcarrete}@inf.uc3m.es

[2]https://github.com/xpn-arcos/xpn

En la evaluación del prototipo inicial de Expand Ad-Hoc se utilizó un cluster heterogéneo con un máximo de 4 nodos de cómputo. Para esta nueva evaluación se va a utilizar el supercomputador MareNostrum 4 con un máximo de 128 nodos de cómputo homogéneos. Lo que nos va a permitir estudiar mejor aspectos como la escalabilidad del sistema de ficheros, entre otros.

Además, para esta nueva evaluación se ha ampliado la batería de pruebas del *benchmark* IOR para incluir una evaluación cuando los ficheros son compartidos por los procesos. También, se ha utilizado el *benchmark* DLIO para evaluar las prestaciones de E/S de Expand Ad-Hoc cuanto se ejecutan aplicaciones de Inteligencia Artificial, dado que este tipo de aplicaciones son muy intensivas en el uso de datos.

El resto del documento se estructura de la siguiente forma: la Sección II muestra el Estado del arte; la Sección III describe las principales características de Expand Ad-Hoc; la Sección IV presenta los resultados de la evaluación de Expand Ad-Hoc en MareNostrum 4. Y, por último, la Sección V contiene las principales conclusiones y los trabajos futuros.

II. Estado del arte

En la actualidad existen diferentes soluciones de almacenamiento, de las cuales, las que están relacionadas con la problemática que se trata en este trabajo son los sistemas de ficheros paralelos y, particularmente, los sistemas de ficheros ad-hoc.

Un sistema de ficheros paralelo es un tipo de sistema de ficheros distribuido que almacena la información de los ficheros entre varios servidores, proporcionando acceso paralelo a estos ficheros [2]. Habitualmente estos sistemas de ficheros utilizan un conjunto de nodos de almacenamiento compartido que se corresponde con un subconjunto del total de nodos que hay en el supercomputador. Esto hace que se generen cuellos de botella en la E/S de datos en estos nodos, lo que provoca que disminuya el rendimiento global de las aplicaciones [3]. Algunos ejemplos conocidos de sistemas de ficheros paralelos son GPFS [4], Lustre [5] y BeeGFS [6].

El sistema de ficheros paralelo GPFS [4] tiene como origen el sistema de archivos Tiger Shark que era un proyecto del Centro de Investigación Almaden de IBM en 1993. Aunque su diseño inicial estaba planeado para aplicaciones multimedia de alto rendimiento, sus características resultaron ser muy adecuadas para la computación científica y, por ello,

es muy común su uso como sistema de ficheros *backend* en los supercomputadores. Una de sus principal característica es que dispone de una caché de datos a nivel de cliente que permite realizar las operaciones de E/S de forma asíncrona y para garantizar la coherencia de datos utiliza un gestor distribuido de cerrojos. Además, GPFS ofrece un acceso paralelo a los ficheros y que permite añadir o eliminar unidades de almacenamiento del clúster GPFS durante su funcionamiento, sin necesidad de realizar más acciones.

Por otro lado, está el sistema de ficheros Lustre [7] que se inició como un proyecto de investigación en la Universidad Carnegie Mellon (CMU) en el año 1999. Tras la aparición de los clúster Beowulf [8] con sistema operativo Linux como alternativa a los costosos supercomputadores, Lustre se convirtió en la solución de almacenamiento para la computación en clúster a gran escala. Además, la expansión de su uso también fue propiciado debido a que utiliza la semántica POSIX y permite accesos concurrentes de lectura y escritura en los ficheros que almacena.

BeeGFS [6] es un sistema de ficheros paralelo que empezó a desarrollarse en 2005 y presentó su primera versión beta en 2007. Está diseñado especialmente para su uso en HPC y también basado en la interfaz POSIX. Además, según los autores, está diseñado para maximizar el rendimiento y la escalabilidad. Para ello, además de dividir los ficheros en bloques y distribuir estos entre todos los nodos de almacenamiento, como hacen el resto de sistemas de ficheros paralelos, también distribuyen los metadatos para balancear la carga.

Un sistema de ficheros ad-hoc permite virtualizar dinámicamente el almacenamiento en los nodos de cómputo teniendo en cuenta los recursos disponibles en los nodos de cómputo del supercomputador y los requerimientos de la aplicación que se va a ejecutar. Esto permite acercar el almacenamiento a la aplicación a través de la localidad de los datos, reduciendo así sobrecarga en el sistema de almacenamiento *backend* del supercomputador que es distribuido y compartido entre todos nodos cómputo [9]. Algunos ejemplos conocidos de sistemas de ficheros ad-hoc son BurstFS [10], GekkoFS [11] y BeeOND [12].

BurstFS [10] es uno de los primeros sistemas de ficheros ad-hoc que se conocen. Está especialmente diseñado y optimizado para ser desplegado como sistema de ficheros local. Sin embargo, este sistema de ficheros tiene el principal inconveniente de que cada vez que se quiere utilizar se tiene que llevar a cabo el despliegue y la inicialización de los servidores de metadatos, requiere de la implementación de mecanismos de redundancia y el cumplimiento total de POSIX [13].

Otro ejemplo de sistemas de ficheros ad-hoc lo constituye GekkoFS [14]. GekkoFS se empezó a desarrollar en el año 2017 y se encuentra actualmente en desarrollo activo. Como características principales, este sistema de ficheros hace uso de la base de datos RocksDB [15] para almacenar los metadatos y la interfaz RPC Mercury [16] para las comunicaciones. Sin embargo, este sistema de ficheros ad-hoc, según describen los autores, no es POSIX completo [11], por lo que algunas llamadas de POSIX no pueden ser utilizadas sobre los ficheros que almacena, como es el caso del *rename*, entre otras. Además, este sistema de ficheros tiene otros problemas relacionados con el uso de múltiples dependencias de otros paquetes de software (RocksDB, Mercury, etc.) y versiones específicas de estos. Lo que provoca que su instalación sea más compleja y su rendimiento dependa de la sobrecarga de dichos componentes y su correcto enlazado con los *drivers* del hardware del supercomputador (por ejemplo, no poder utilizar los *drivers* de Omni-Path u otras implementaciones de InfiniBand).

Por último, está BeeOND [12] que es otro sistema de ficheros ad-hoc que permite desplegar diferentes instancias del sistema de ficheros BeeGFS [6] de forma dinámica, aprovechando las ventajas de este. Esto puede ser útil en escenarios como la nube o cuando se tiene como sistema de ficheros *backend* en el supercomputador BeeGFS. Aunque para uso personal puede ser gratuito, para su uso profesional BeeOND precisa de pago, a diferencia de Expand Ad-Hoc o GekkoFS.

III. Expand Ad-Hoc

Expand Ad-Hoc [1] es un sistema de ficheros paralelo basado en servidores ad-hoc diseñado especialmente para la ejecución de flujos de trabajo de aplicaciones intensivas en el uso de datos. A continuación, se van a describir brevemente sus características principales.

A. Diseño de Expand Ad-Hoc

El diseño de este sistema de ficheros, como se puede ver en la Figura 1, se basa en el uso de un conjunto de servidores de datos y clientes ad-hoc se comunican entre sí utilizando MPI, lo que facilita su uso en los entornos HPC.

Estos clientes y servidores ad-hoc pueden ser desplegados en los nodos de cómputo donde ejecuta la aplicación, permitiendo aprovechar la localidad de los datos, o en nodos de cómputo diferentes.

Además, cabe destacar que Expand Ad-Hoc utiliza el almacenamiento local de los nodos de cómputo (HDD, SSD, memoria compartida, etc.) lo que permite utilizar los servicios de gestión de datos proporcionados por el sistema operativo, como puede ser POSIX.

B. Distribución de datos y archivos

Expand Ad-Hoc virtualiza el almacenamiento local de los nodos de cómputo generando una o varias particiones genéricas paralelas. Para ello, cada nodo de cómputo que ejecuta un servidor ad-hoc proporciona uno o más directorios que se combinan para generar estas particiones distribuidas.

En cuanto al almacenamiento de los ficheros, como se puede ver en la Figura 2, estos, en primer lugar, son divididos en bloques del mismo tamaño, siendo

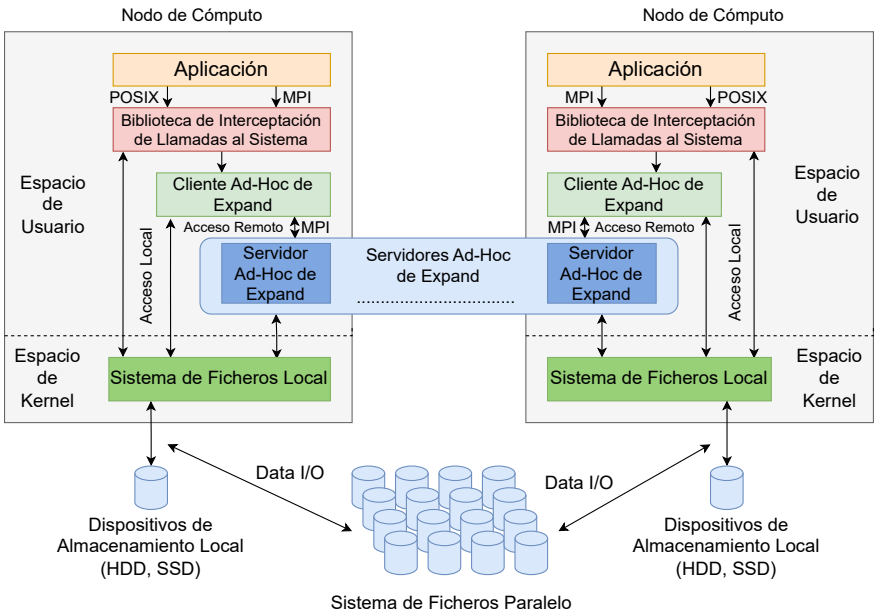

Fig. 1: Diseño de la arquitectura de XPN Ad-Hoc.

este tamaño el *blocksize* de la partición de Expand Ad-Hoc donde se van a almacenar. Después, se distribuyen dichos bloques equitativamente entre todos los servidores ad-hoc que forman la partición virtual utilizando el patrón de reparto *round-robin* (de forma análoga a un RAID 0 de discos) generando un subfichero por nodo de cómputo. Esta metodología de almacenamiento facilita el acceso paralelo a ficheros y es totalmente transparente para los usuarios de Expand Ad-Hoc.

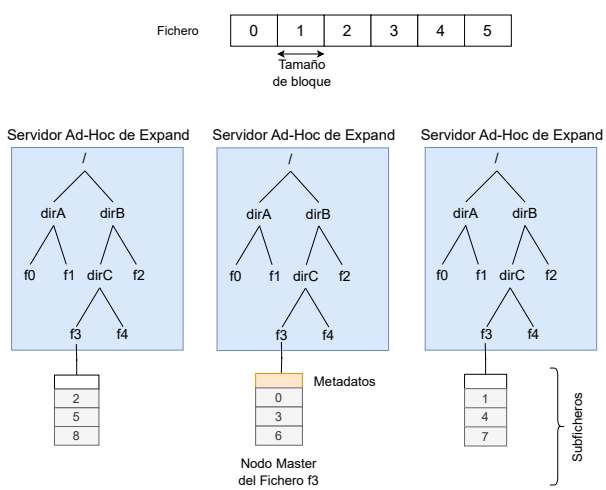

Fig. 2: Estructura de archivos y la asignación de directorios en Expand.

C. Gestión de metadatos

Respecto a la gestión de los metadatos de los ficheros, Expand Ad-Hoc no utiliza ningún tipo de gestor de metadatos, si no que cada uno de los subficheros tiene una pequeña cabecera al comienzo.

Sin embargo, a pesar de que todos los subficheros tengan este espacio reservado al comienzo para guardar metadatos, solo el subfichero almacenado en el nodo maestro de la partición contendrá esta información (ver Figura 2). Para determinar cuál es el nodo maestro entre todos los que forman parte de la partición, se convierte el nombre del fichero en un número aplicando sobre el nombre una función *hash*. De esta forma se reparte la carga de acceder al nodo maestro entre todos los nodos de almacenamiento disponibles.

D. Acceso paralelo

Expand Ad-Hoc hace uso de un manejador de fichero virtual para llevar a cabo todas las operaciones los ficheros que almacena. Esto permite optimizar las operaciones de E/S, ya que se dividen las operaciones de los usuarios en varias operaciones que se llevan a cabo de forma paralela en los servidores ad-hoc involucrados.

E. Localidad de datos

Como se ha indicado en las líneas anteriores, los servidores ad-hoc de Expand Ad-Hoc pueden ser desplegados en los mismos nodos de cómputo en los que van a ejecutar las aplicaciones del *workflow*. Esto permite aprovechar la localidad de los datos cuando los datos a los que desea acceder la aplicación se encuentran almacenados en el servidor ad-hoc que ejecuta en este mismo nodo de cómputo.

Cuando se da esta situación, Expand Ad-Hoc es capaz de detectarlo y acceder a los datos directamente, como se puede ver en la Figura 1, utilizando el sistema de ficheros local, sin tener que comunicarse con el servidor de datos. Lo que permite optimizar el acceso a los datos, ya que se evita la latencia y sobrecarga que se producen con los accesos remotos a datos.

F. Biblioteca de interceptación de llamadas al sistema para las aplicaciones POSIX ya existentes

Por último, Expand Ad-Hoc ofrece una biblioteca de interceptación de llamadas al sistema que permite utilizar este sistema de ficheros sin tener que modificar el código fuente de las aplicaciones ya existentes, que en muchas ocasiones no es posible.

Esta biblioteca intercepta la mayoría de las llamadas al sistema POSIX que se realizan a lo largo de la ejecución de una aplicación. Cuando se intercepta una llamada al sistema, la biblioteca, en primer lugar, comprueba que el fichero con el que se trabaja esté almacenado en Expand Ad-Hoc, en cuyo caso se llamará a la función correspondiente de la API de este sistema de ficheros. Sin embargo, si el fichero no se encuentra almacenado en Expand Ad-Hoc se ejecutará la llamada al sistema de la `libc.so` correspondiente.

Para hacer uso de esta biblioteca de interceptación se utiliza `LD_PRELOAD` que permite cargar esta biblioteca antes que el resto, incluida la `libc.so`, sin necesidad de permisos de superusuario.

IV. Evaluación

Con la evaluación presentada en esta Sección, por un lado, se pretende medir el rendimiento ofrecido por Expand Ad-Hoc cuando se realizan escrituras y lecturas de fichero por una aplicación paralela, y también el ancho de banda de E/S obtenido durante la ejecución de un entrenamiento de *Deep Learning*. Por otro lado, en esta Sección se compara el rendimiento de Expand Ad-Hoc respecto a GPFS, que es un sistema ficheros paralelo que habitualmente se utiliza en los supercomputadores como sistema de ficheros *backend*.

Para esta evaluación se ha decidido utilizar, en primer lugar, el *benchmark* de código abierto IOR [3]. Este *benchmark* es muy utilizado para la evaluación de los sistemas de ficheros paralelos y distribuidos ya que permite simular diferentes cargas de E/S (operaciones de lectura/escritura, acceso a fichero compartido/individual, etc.) y obtener el ancho de banda ofrecido por el sistema de ficheros.

Además, también se ha utilizado el *benchmark* desarrollado por Argonne Leadership Computing Facility DLIO [4] [17], que permite medir el rendimiento de un sistema de ficheros a través de la emulación del comportamiento de E/S de las aplicaciones científicas de *Deep Learning*.

Para llevar a cabo esta evaluación se ha utilizado el supercomputador MareNostrum 4 [18], que dispone de 3456 nodos de cómputo Lenovo ThinkSystem SD530 distribuidos en 48 racks. Concretamente cada nodo dispone de:

- Dos procesadores Intel Xeon Pentium 8160 24C con 24 procesadores de 2.1 GHz.
- Un Intel SSD DC S3520 Series con 240 GiB de almacenamiento disponible.
- 96 GiB (2 GiB per core) de memoria principal.
- Una red de 100 Gb Intel Omni-Path Full-Fat Tree.

Por lo que, en total este supercomputador dispone de 165888 procesadores y 384.75 TiB de memoria principal.

Para la evaluación realizada en este trabajo, se ha usado la configuración por defecto recomendada en cada uno de los sistemas de ficheros. En el caso del almacenamiento:

- Expand Ad-Hoc está configurado con un tamaño de bloque de 512 KiB, lo que permite comparar ambos sistemas de ficheros en las mismas condiciones.
- Expand Ad-Hoc usa el dispositivo SSD y la memoria compartida (SHM) disponible en cada uno de los nodos como almacenamiento subyacente. Para la memoria compartida (SHM) en ambos casos se usa el directorio `/dev/shm` como sistema de ficheros.

En el caso de comunicación en red:

- Expand Ad-Hoc usa la configuración por defecto de MPI en el supercomputador (sin ningún parámetro adicional).

A. Evaluación con IOR

Para evaluar Expand Ad-Hoc con el *benchmark* IOR, se han ejecutado diferentes pruebas de rendimiento para obtener el ancho de banda de escritura y lectura en accesos paralelos a un fichero. Además, las pruebas de rendimiento realizadas en este trabajo se basan en las realizadas en [19] para la evaluación de GekkoFS.

Las configuraciones que se han usado para las evaluaciones con IOR son las siguientes:

- Nodos de cómputo: 1, 2, 4, ... hasta un total de 128 nodos.
- Almacenamiento local: SSD y memoria compartida (SHM).
- Tamaño de transferencia usado en IOR: 64 KiB, 512 KiB y 1 MiB.
- Procesos cliente por nodo de computación: 8.
- Operaciones: lectura y escritura en paralelo sobre un fichero compartido y un fichero por proceso.
- Tamaño escrito por cada proceso cliente: 4 GiB (lo que supone un total de 4 TiB de fichero en la configuración máxima).
- Sistemas de ficheros: GPFS 4.2.2.0 y Expand Ad-Hoc 2.2.2.

Por último, cabe destacar que para todas las pruebas realizadas se ha utilizado un servidor Expand Ad-Hoc por nodo de cómputo (de 1 hasta 128). Y cada una de las pruebas se han ejecutado como trabajo individual con los nodos de cómputo en exclusividad mediante el sistema de colas SLURM (*Simple Linux Utility for Resource Management*) [20].

Las Figuras 3 y 4 muestran los resultados obtenidos en la evaluación llevada a cabo utilizando IOR

[3] https://github.com/hpc/ior
[4] https://github.com/argonne-lcf/dlio_benchmark

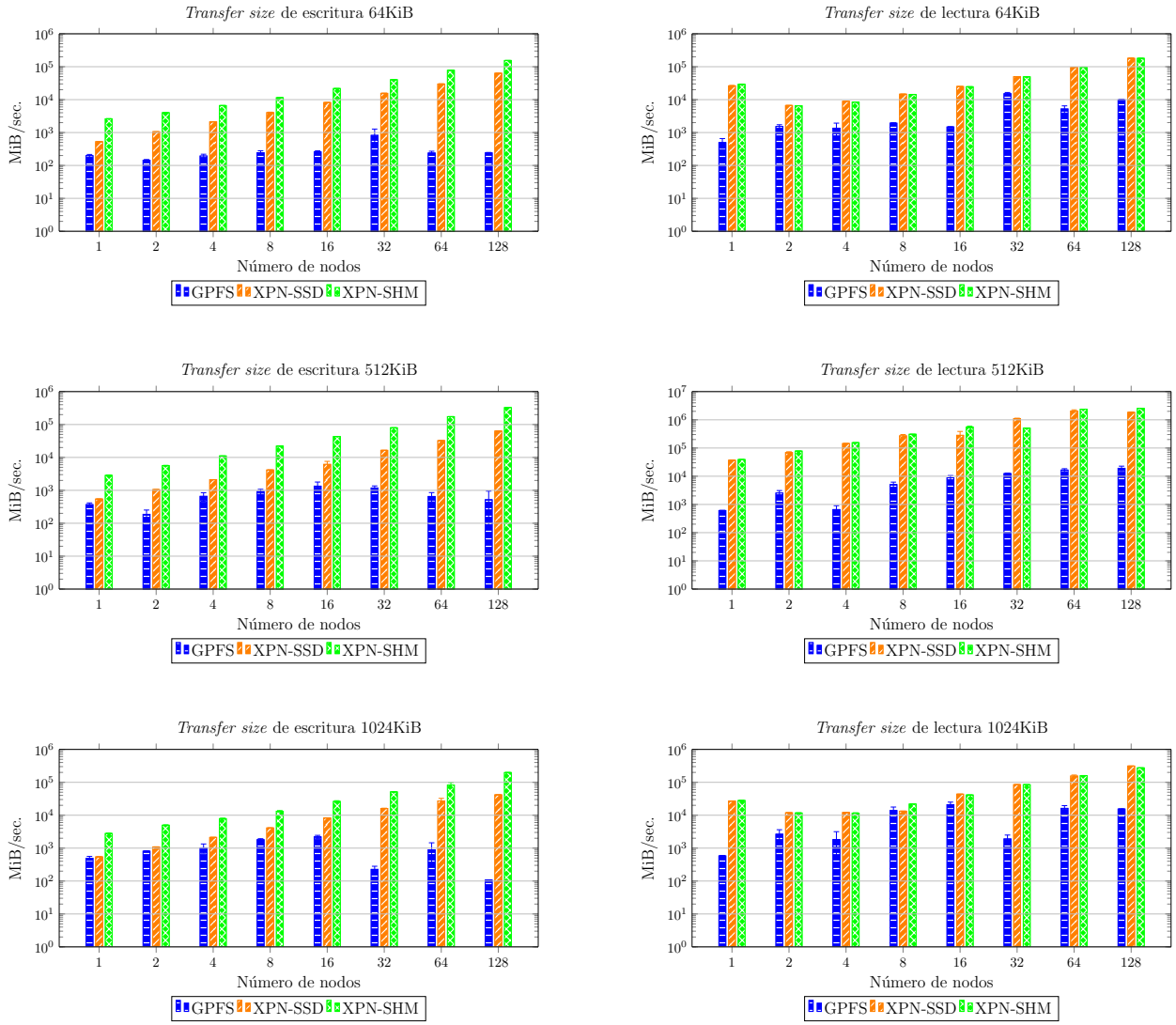

Fig. 3: GPFS vs. Expand Ad-Hoc. Ancho de banda (MiB/seg.) escribiendo y leyendo con diferentes tamaños de transferencia (64KiB, 512KiB y 1MiB), nodos de cómputo (1, 2, 4, 8, 16, 32, 64 y 128), con 8 procesos cliente por nodo y **fichero compartido**. Resultados en escala logarítmica.

con un fichero compartido y un fichero por proceso, respectivamente. Concretamente, se muestran el ancho de banda (MiB/segundo en escala logarítmica) cuando se escribe y leen datos usando como tamaños de transferencia (*transfer sizes*) 64 KiB, 512 KiB y 1024 KiB en GPFS, así como Expand Ad-Hoc con SSD (XPN Ad-Hoc - SSD) y con memoria compartida (XPN Ad-Hoc - SHM).

En los resultados de rendimiento obtenidos con el *benchmark* IOR se puede observar que cuando se utiliza un fichero compartido (ver Figura 3) entre todos los procesos, Expand Ad-Hoc usando un disco SSD obtiene mucho mejor rendimiento de escritura y lectura que GPFS. En el caso de utilizar memoria compartida como almacenamiento local en Expand Ad-Hoc, se obtienen los mejores resultados.

En cuanto a los resultados obtenidos cuando se utilizan ficheros individuales por proceso (ver Figura 4), el sistema de ficheros GPFS ofrece en escrituras un ancho de banda mayor que los sistemas de ficheros ad-hoc. Este comportamiento se debe a que, como se ha comentado anteriormente, GPFS dispone de una caché de datos para realizar de forma asíncrona las operaciones de E/S, mientras que en el caso de Expand Ad-Hoc las operaciones se realizan directamente en disco. Además, al tratarse de ficheros individuales (sin que exista concurrencia) se evita la sobrecarga del servicio de cerrojos distribuido. No obstante, cuando el número de nodos de cómputo incrementa el ancho de banda de ambos sistemas de ficheros la diferencia disminuye. Esto se debe a que Expand Ad-Hoc también incrementan el número de servidores de almacenamiento, permitiendo una mejor escalabilidad. Por otro lado, en el caso de las lecturas Expand Ad-Hoc obtiene mejores anchos de banda que el resto en la gran mayoría de los casos.

Como resumen de la evaluación llevada a cabo con el *benchmmark* IOR, se puede afirmar que Expand Ad-Hoc globalmente obtiene mejores resultados que GPFS ya que al utilizarse el almacenamiento local de los nodos permite escalar el número de servidores de almacenamiento y, además, aprovechar la localidad de los datos. Todo esto se consigue gracias al uso de MPI, que permite aprovechar todo el rendimiento de la red, y de dejar que las aplicaciones sean las encargadas de aplicar un protocolo de coherencia si lo

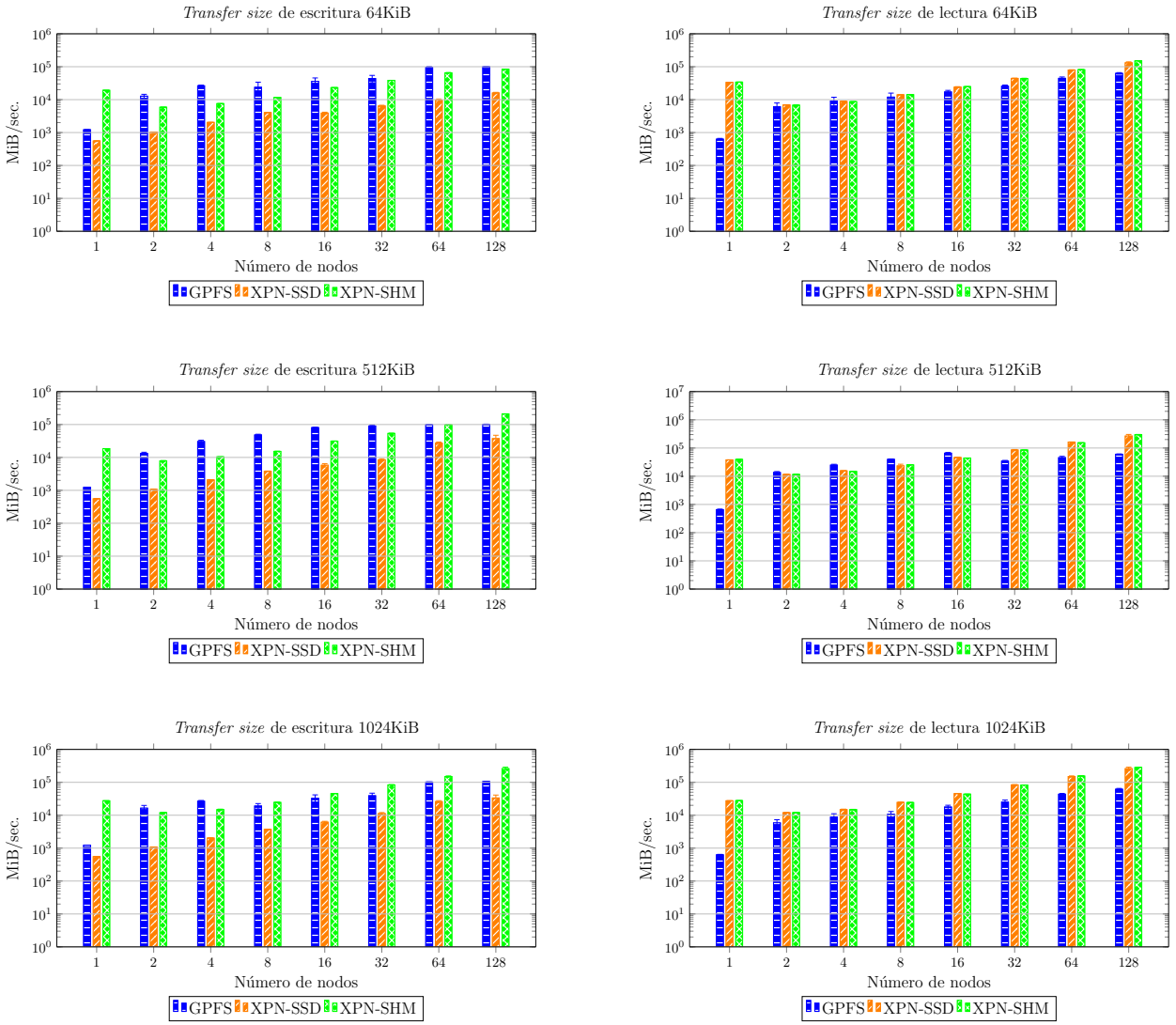

Fig. 4: GPFS vs. Expand Ad-Hoc. Ancho de banda (MiB/seg.) escribiendo y leyendo con diferentes tamaños de transferencia (64KiB, 512KiB y 1MiB), nodos de cómputo (1, 2, 4, 8, 16, 32, 64 y 128), con 8 procesos cliente por nodo y un **fichero individual por proceso**. Resultados en escala logarítmica.

necesitan para evitar esta sobrecarga cuando no sea necesario. Por ejemplo, al usar ficheros compartidos en los que cada proceso escribe en una parte distinta del fichero, es decir, ningún bloque del fichero es modificado por más de un proceso.

B. Evaluación con DLIO

Por otro lado, se ha querido evaluar el rendimiento que ofrece Expand Ad-Hoc cuando se utiliza este con aplicaciones de *Deep Learning*. Para ello, como se mencionó anteriormente, se ha utilizado el *benchmark* DLIO con diferentes cargas de trabajo (*workloads*).

Las configuraciones que se han utilizado para estas evaluaciones son:

- Nodos de cómputo: 16, 32 y 64 nodos.
- Almacenamiento local: SSD y memoria compartida (SHM).
- Cargas de trabajo (con su configuración por defecto):
  - CosmoFlow: red CNN 3D para estudiar el universo. Tamaño del *dataset*: 65 GiB. Número de *epoch*: 4.
  - ResNet50: clasificación de imágenes 3D. Tamaño del *dataset*: 147 GiB. Número de *epoch*: 1.
  - UNET3D: segmentación de imágenes médicas 3D. Tamaño del *dataset*: 36 GiB. Número de *epoch*: 10.
- Sistemas de ficheros: GPFS 4.2.2.0 y Expand Ad-Hoc 2.2.2.

Al igual que en la evaluación llevada a cabo con el *benchmark* IOR, se ha utilizado un servidor Expand Ad-Hoc por cada nodo de cómputo y, también, se han utilizado estos nodos de cómputo en exclusividad utilizando el sistema de colas SLURM.

En la Figura 5 se puede ver el ancho de banda (MiB/segundo) de E/S obtenido durante el entrenamiento realizado por DLIO para cada una de las cargas de trabajo sobre GPFS y Expand Ad-Hoc utilizando como almacenamiento local SSD (XPN Ad-Hoc - SSD) y memoria compartida (XPN Ad-Hoc - SHM).

En los resultados, se puede ver que el ancho de

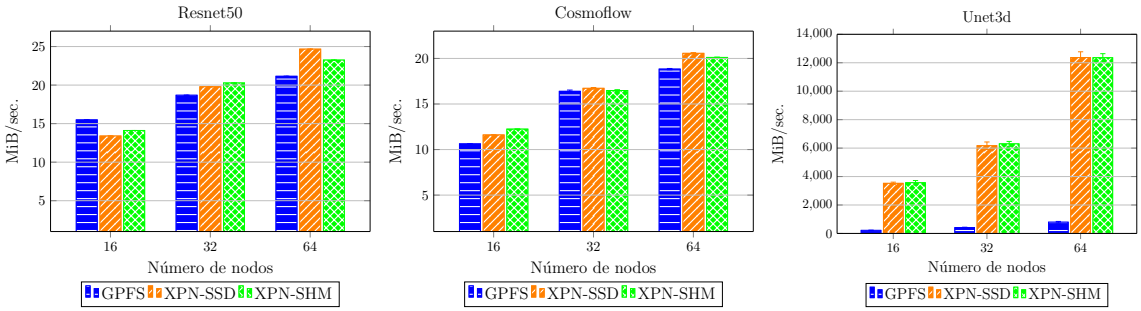

Fig. 5: GPFS vs. Expand Ad-Hoc. Ancho de banda (MiB/seg.) del entrenamiento realizado por DLIO y 16, 32 y 64 nodos de cómputo.

banda obtenido por el sistema de ficheros Expand Ad-Hoc es ligeramente mayor, de forma general, que el ofrecido por GPFS cuando se ejecutan las cargas de trabajo CosmoFlow y ResNet50, mientras que es mucho mayor cuando se ejecuta el *workload* UNET3D.

Este comportamiento se debe a que Expand Ad-Hoc aprovecha la localidad de los datos, ya que en el caso de UNET3D se realizan 10 *epoch* sobre el mismo *dataset*, mientras que en el caso de CosmoFlow y ResNet50 se realizan 4 y 1 *epoch*, respectivamente. Es decir, Expand Ad-Hoc está diseñado para utilizar el almacenamiento local de los nodos de cómputo para guardar los datos que van a necesitar estos nodos de cómputo durante la ejecución de una aplicación. Esto evita tener que obtener los datos del sistema de almacenamiento *backend*, procedimiento que, de forma general, es más costoso ya que este sistema de ficheros es compartido por todos los nodos de cómputo y son accesos remotos. Por lo tanto, cuanto mayor es el número de *epochs* realizados, mayor es el número de accesos evitados al sistema de almacenamiento *backend* ya que se realizan de forma local en los propios nodos, de ahí a la mejora de rendimiento obtenida en UNET3D respecto a GPFS.

V. Conclusiones

En este artículo se ha presentado el sistema de ficheros paralelo Expand Ad-Hoc para entornos HPC y su evaluación sobre el supercomputador MareNostrum 4 utilizando los *benchmarks* IOR y DLIO.

Como se ha podido ver en las evaluaciones presentadas en este trabajo, el diseño de Expand Ad-Hoc utilizando servidores de datos basados en MPI y el aprovechamiento de la localidad permite obtener una mayor escalabilidad y un mayor ancho de banda cuando se realizan operaciones de E/S sobre uno o más ficheros de forma simultánea respecto a un sistema de ficheros paralelo utilizado como *backend*, como es el caso de GPFS.

Como principales líneas de trabajos futuros se tiene el diseño de soporte de maleabilidad, así como su evaluación de rendimiento. Además, también se quiere dotar a Expand Ad-Hoc de tolerancia a fallos basada en replicación. También, se quiere probar Expand Ad-Hoc con el *benchmark* DLIO incrementando el número de *epochs* en las diferentes cargas de trabajo, así como probar Expand Ad-Hoc con aplicaciones reales.

Agradecimientos

Este trabajo ha sido parcialmente apoyado por el proyecto del Ministerio de Ciencia e Inoovación español “Expand: High Performance Storage System for HPC and Big Data Environments”. Ref. TED2021-131798B-I00.

También parcialmente financiado por el proyecto *Adaptive multi-tier intelligent data manager for Exascale (ADMIRE)* del programa *European Union’s Horizon 2020 JTI-EuroHPC research and innovation programme*. Ref. H2020-JTI-EuroHPC-2019-1, Project no. 956748.

Y parcialmente apoyado por el proyecto EuroHPC “Adaptive multi-tier intelligent data manager for Exascale” bajo la subvención 956748 - ADMIRE - H2020-JTI-EuroHPC-2019-1 y por la Agencia Española de Investigación bajo la subvención PCI2021-121966.

Los autores agradecen los recursos informáticos de MareNostrum 4 y el apoyo técnico prestado por el Barcelona Supercomputing Center (IM-2023-1-0012).

Referencias

[1] Diego Camarmas-Alonso, Felix Garcia-Carballeira, Alejandro Calderon Mateos, and Jesus Carretero Perez, “Sistema de almacenamiento Ad-Hoc para entornos HPC basado en el sistema de ficheros paralelo Expand,” in *XXXII Jornadas de Paralelismo (JP21/22)*. July 2022, Disponible en Zenodo.

[2] Viacheslav Dubeyko, “Comparative analysis of distributed and parallel file systems’ internal techniques,” 2019.

[3] Bing Xie, Sarp Oral, Christopher Zimmer, Jong Youl Choi, David Dillow, Scott Klasky, Jay Lofstead, Norbert Podhorszki, and Jeffrey S. Chase, “Characterizing output bottlenecks of a production supercomputer: Analysis and implications,” *ACM Trans. Storage*, vol. 15, no. 4, jan 2020.

[4] Frank Schmuck and Roger Haskin, “GPFS: A Shared-Disk file system for large computing clusters,” in *Conference on File and Storage Technologies (FAST 02)*, Monterey, CA, Jan. 2002, USENIX Association.

[5] Jaehyun Han, Deoksang Kim, and Hyeonsang Eom, “Improving the performance of lustre file system in hpc environments,” in *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016, pp. 84–89.

[6] Frank Herold, Sven Breuner, and Jan Heichler, “An introduction to beegfs,” *ThinkParQ, Kaiserslautern, Germany, Tech. Rep*, 2014.

[7] Peter Braam, “The lustre storage architecture,” *CoRR*, vol. abs/1903.01955, 2019.

[8] Daniel A. Reed, “Beowulf clusters: From research curiosity to exascale,” in *Proceedings of the 20 Years of*

Beowulf Workshop on Honor of Thomas Sterling's 65th Birthday, New York, NY, USA, 2014, Beowulf '14, p. 28–33, Association for Computing Machinery.

[9] André Brinkmann, Kathryn Mohror, Weikuan Yu, Philip Carns, Toni Cortes, Scott A Klasky, Alberto Miranda, Franz-Josef Pfreundt, Robert B Ross, and Marc-André Vef, "Ad hoc file systems for high-performance computing," *Journal of Computer Science and Technology*, vol. 35, no. 1, pp. 4–26, 2020.

[10] Teng Wang, Kathryn Mohror, Adam Moody, Kento Sato, and Weikuan Yu, "An ephemeral burst-buffer file system for scientific applications," in *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2016, pp. 807–818.

[11] Marc-André Vef, Nafiseh Moti, Tim Süß, Markus Tacke, Tommaso Tocci, Ramon Nou, Alberto Miranda, Toni Cortes, and André Brinkmann, "Gekkofs: A temporary burst buffer file system for hpc applications," *Journal of Computer Science and Technology*, vol. 35, no. 1, pp. 72–91, 2020.

[12] Zoya Masih, "On demand file systems with beegfs," 2023.

[13] IEEE-SA Standards Board, "Ieee standard for information technology-portable operating system interface (posix)-part 1: System application program interface (api)- amendment d: Additional real time extensions [c language]," Tech. Rep., IEEE, 1999.

[14] Marc-André Vef, Nafiseh Moti, Tim Süß, Tommaso Tocci, Ramon Nou, Alberto Miranda, Toni Cortes, and André Brinkmann, "Gekkofs-a temporary distributed file system for hpc applications," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 319–324.

[15] Siying Dong, Andrew Kryczka, Yanqin Jin, and Michael Stumm, "Rocksdb: Evolution of development priorities in a key-value store serving large-scale applications," *ACM Trans. Storage*, vol. 17, no. 4, oct 2021.

[16] Jerome Soumagne, Dries Kimpe, Judicael Zounmevo, Mohamad Chaarawi, Quincey Koziol, Ahmad Afsahi, and Robert Ross, "Mercury: Enabling remote procedure call for high-performance computing," in *2013 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2013, pp. 1–8.

[17] H. Devarajan, H. Zheng, A. Kougkas, X.-H. Sun, and V. Vishwanath, "Dlio: A data-centric benchmark for scientific deep learning applications," *21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, , no. 81–91, 2021.

[18] BSC, "MareNostrum specification," 2023, Accessed Jan. 2, 2023. [Online].

[19] Marc-Andre Vef, Nafiseh Moti, Tim Süß, Markus Tacke, Tommaso Tocci, Ramon Nou, Alberto Miranda, Toni Cortes, and André Brinkmann, "GekkoFS - A temporary burst buffer file system for HPC applications," *J. Comput. Sci. Technol.*, vol. 35, no. 1, pp. 72–91, 2020.

[20] SLURM, "Slurm workload manager," 2023, Accessed Jan. 2, 2023. [Online].