Análisis funcional de la pila de software de E/S paralela utilizando *IaaS*

Diego Encinas[1], Sandra Mendez[2], Dolores Rexachs[3], Emilio Luque[3], Marcelo Naiouf[1], and Armando De Giusti[1]

[1]*Informatics Research Institute LIDI, CIC's Associated Research Center. National University of La Plata, 50 y 120, La Plata, 1900, Argentina.*
{dencinas, mnaiouf, degiusti}@lidi.info.unlp.edu.ar
[2]*High Performance Systems Division, Leibniz Supercomputing Centre (LRZ), Boltzmannstr. 1, Garching near Munich, 85748, Germany*
sandra.mendez@lrz.de
[3]*Computer Architecture and Operating Systems Department. Universitat Autònoma de Barcelona. Edifici Q Campus UAB. Bellaterra, 08193, Spain*
{dolores.rexachs, emilio.luque}@uab.es

Abstract

Las operaciones de Entrada/Salida en los centros de supercomputación siguen siendo un cuello de botella para el procesamiento en paralelo. Esto provoca que la capacidad de procesamiento de los distintos nodos que componen estos centros se vea limitada. Por este motivo, en el presente trabajo se propone estudiar y evaluar el rendimiento de las principales operaciones paralelas de E/S. Las ejecuciones se realizan utilizando dos tipos de clústeres (físico y virtual), empleando benchmarks específicos de E/S y sistemas de archivos. Además, se propone diseñar, desarrollar e incorporar pequeñas secciones de código a algunas de las funciones más importantes invocadas durante la ejecución de los benchmarks, de manera de obtener medidas de tiempo para cuantificar la ejecución en los distintos puntos del sistema de software de E/S.

Keywords: Cloud Computing, Pila de Software de E/S, PVFS2, Instrumentación de Código

1 Introducción

Los actuales sistemas de HPC proporcionan cada vez mayor capacidad de cómputo debido al rápido avance de las tecnologías de procesadores. Ese avance no es proporcional en el área de almacenamiento. Para reducir la diferencia entre el rendimiento computacional y el de almacenamiento, los actuales sistemas de E/S utilizan hardware de alto rendimiento por medio de redes de almacenamiento de alta velocidad y sistemas de discos en RAID para proporcionar E/S paralela. Para gestionar el hardware de E/S se usa la pila de software que está compuesta por capas que gestionan los diferentes módulos de la arquitectura.

La Figura 1 muestra dos visiones del sistema de E/S, la infraestructura hardware y de la pila de software, el rendimiento del sistema depende de la carga de trabajo (patrones de E/S de las aplicaciones) y de la configuración del sistema. Existe una gran variabilidad en el comportamiento global del sistema dependiendo de la configuración y de los patrones de acceso a la E/S generados por las aplicaciones, observar el impacto de la configuración del sistema requiere el uso de sistemas rápidamente reconfigurables [1].

Una operación de E/S debe atravesar la estructura de la pila de software para escribir/leer a/desde los dispositivos. El encargado de gestionar el acceso a los datos es el sistema de ficheros, en HPC para conseguir el paralelismo a nivel de E/S, se usan sistemas de ficheros paralelos. Cuando una aplicación se está ejecutando y realiza una operación de E/S, las librerías de E/S reciben la operación e interaccionan con el sistema de ficheros. Este tipo de sistemas multicapas trae una complejidad adicional que está relacionado con el análisis de rendimiento y de funcionalidad de toda la pila de software.

Una de las técnicas para evaluar el rendimiento del Sistema de E/S a nivel de libreria es el uso de *benchmarks* específicos como IOR, BT-IO, Madbench2, entre otros que generan diferentes patrones de acceso[2]. Para poder explicar el rendimiento observado se pueden utilizar herramientas de análisis de rendimiento para HPC, pero la mayoria de las herramientas que permiten analizar el rendimiento de la E/S lo hacen a nivel MPI-IO y de POSIX-IO. Por lo cual no es posible obtener información sobre cómo interactúan todas las capas de la pila de SW de E/S.

Entender la interaccion entre las capas de la pila de software es importante no solo para explicar el comportamiento observado, sino tambien para detectar cuellos de botellas y proponer tecnicas de optimizacion. Ademas, este tipo de informacion es fundamental para modelar y simular configuraciones del sistema de E/S para poder evaluar el impacto de las diferentes capas

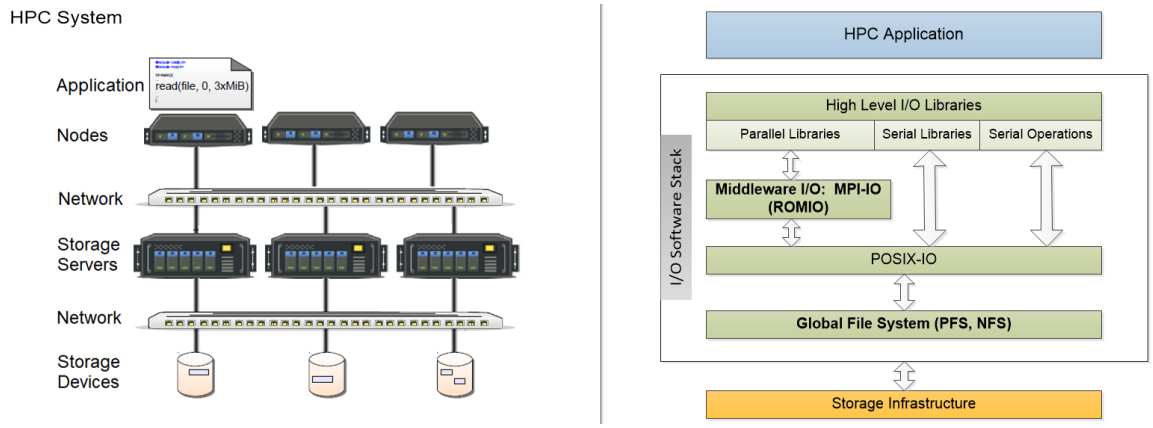

Figure 1: Sistema de HPC: infraestructura y pila de software de E/S

sin alterar sistemas en produccion.

En este artículo se presenta un análisis funcional de pila de software de E/S, por medio de la instrumentación de código en las capas de MPI-IO y del sistema de ficheros paralelo utilizando el servicio IaaS de Amazon. Para identificar las diferentes funciones que interactúan se ha optado por una técnica de instrumentación de código fuente en la capa MPI-IO y del sistema de fichero paralelo. Se ha seleccionado una plataforma Cloud, para desplegar un clúster de HPC y configurar la pila de software con todas las modificaciones necesarias. Se ha optado por una plataforma IaaS debido a que los sistemas de I/O en clusters fisicos requieren permisos de administrador y cualquier cambio afecta a todos los usuarios del cluster. Además, se pueden evaluar varias configuraciones de E/S de manera más rápida y practica sin afectar el normal funcionamiento de un sistema de HPC en producción. El trabajo está organizado de la siguiente manera, en la Sección 2 se presentan los conceptos relacionados al software utilizado. En la Sección 3 se describen las capas de la pila de software elegidas para explicar el trabajo. La Sección 4 muestra los experimentos realizados y los resultados encontrados. Finalmente, en la Sección 5 se presentan las conclusiones y trabajos futuros.

2 Conceptos relacionados

En esta sección se introducen algunos conceptos básicos relacionados a la pila de software de E/S.

2.1 Middleware

La Interfaz de Pasaje de Mensajes (MPI) es un sistema estandarizado y portable diseñado por un grupo de investigadores tanto del sector académico como del industrial para ser utilizado en una amplia variedad de arquitecturas de cómputo paralelo. Mientras que MPI-IO fue desarrollado en el Laboratorio Watson de IBM para proveer soporte de E/S paralela para MPI.

Los llamados de funciones de MPI-IO son muy similares a los de MPI, escribir archivos MPI es semejante a enviar mensajes MPI y leer archivos MPI es parecido a recibir mensajes MPI. En este trabajo se utilizó MPICH que es una implementación de alto rendimiento y ampliamente portátil del estándar MPI. MPICH [3] está siendo actualmente distribuido como software abierto, con una licencia libre y gratuita. Ha sido probado en múltiples plataformas, incluyendo Linux (en IA32 y x86-64), Mac OS/X (PowerPC e Intel), Solaris (32 y 64 bits) y Windows.

2.2 Benchmarks

Los benchmarks están diseñados para imitar un tipo particular de carga de trabajo en un componente o sistema. Los benchmarks sintéticos lo hacen creando programas especiales para imponer la carga de trabajo en el componente, mientras que los benchmarks de aplicación ejecutan programas del mundo real sobre el sistema. Los benchmarks de aplicación suelen ofrecer una mejor medición del rendimiento real de un sistema determinado, pero los sintéticos son útiles para probar componentes individuales, como un disco duro o un dispositivo de red. Idealmente, los benchmarks sólo deberían sustituir a las aplicaciones reales si la aplicación no está disponible o es demasiado difícil o costosa de ejecutar en un procesador o sistema determinado.

Se ha optado por dos de los benchmarks de E/S más populares para evaluar los rendimientos de las operaciones paralelas de E/S en las que se centrará este trabajo: uno sintético, conocido como IOR, y otro de aplicación, llamado BTIO.

2.3 Sistemas de Archivos Paralelos

Al constituir un clúster de cómputo de alto rendimiento, se tienen tres categorías principales de sistemas de archivos: NFS, los sistemas de red de área de almacenamiento (SAN) y los sistemas de archivos paralelos. El primero suele ser considerado fácil de

utilizar, pero el servidor NFS puede ser un lugar de fallo crítico y NFS no es correctamente escalable a través de grandes clústeres.

En cambio, entre las principales ventajas que puede proveer un sistema de archivos paralelo se puede mencionar un espacio de nombre global, escalabilidad y la capacidad de distribuir grandes archivos a través de múltiples nodos en un entorno de clúster, haciendo que un sistema de archivos de esta clase sea muy apropiado para subsistemas de E/S en HPC. Generalmente, un sistema de archivos paralelo implica un servidor de metadatos que contiene información sobre los datos en los nodos de E/S. Algunos sistemas utilizan un servidor particular para los metadatos, mientras otros distribuyen la funcionalidad del servidor de metadatos a través de los nodos de E/S. Algunos ejemplos de sistemas de archivos paralelos para clústeres de cómputo de alto rendimiento son PVFS y Lustre.

En este trabajo se ha optado por PVFS2 que es soportada por la Universidad Clemson y el Laboratorio Argonne. Aunque, puede mencionarse que Clemson desarrolló en 2008 una nueva rama llamada OrangeFS que incluye extensiones para soportar grandes directorios de pequeños archivos, mejoras de seguridad y capacidades de redundancia. Hacia el año 2011 OrangeFS se convirtió en la línea de desarrollo principal, aunque la mayoría de las actualizaciones son aplicables a ambas ramas [4].

3 Análisis funcional de la pila de software

Desde la petición por parte de la aplicación de usuario hasta que la misma llega a realizarse físicamente sobre el disco, es importante determinar los programas y módulos que utilizan cada una de las operaciones, esto permitirá decidir dónde incluir código adicional para determinar el tiempo de ejecución de las distintas funciones ante las diferentes operaciones de E/S. En la Figura 2 se muestra un esquema de la estructura analizada para realizar lo antes dicho.

En la Figura 2 se puede ver la estructura interna de MPICH con respecto a su implementación de MPI-IO y su vínculo con el sistema de archivos del cluster. ROMIO es una implementación portable y de alto rendimiento de MPI-IO, optimizada para patrones de acceso no contiguos, los cuales son comunes en las aplicaciones paralelas. Un componente clave de ROMIO, que permite una implementación tan portable de MPI-IO, es una capa interna de dispositivo abstracto para E/S paralela llamada ADIO (Abstract-Device Interface for I/O). Es el componente `ADIO` el que contiene los módulos que incluyen las funciones que envían las peticiones de E/S de las aplicaciones de usuario hacia el sistema de archivos paralelo. Es decir, la capa ADIO Common contienen las funciones comunes a todos los sistemas de archivos. Mientras que `AD_PVFS` contienen las funciones para implementar la interfaz MPI-IO en el sistema de archivos PVFS2.

3.1 Instrumentación

La instrumentación insertada en el código de los componentes de MPICH y del sistema de ficheros PVFS2 [5] ha sido utilizada para identificar las funciones invocadas durante la ejecución de una aplicación paralela. En particular las interacciones de alguna funciones que son ejecutadas durante el procesamiento de determinadas solicitudes como son las de apertura / cierre, escritura y lectura de archivos.

En la Figura 3 se muestran, por ejemplo, las funciones en la capa de *ROMIO* y de *System Interface* de PVFS2, detectadas durante la ejecucion de *IOR* y *BT-IO clase FULL.*

4 Evaluación Experimental

Plataformas como Amazon ofrecen distintos tipos de instancias de acuerdo al tipo de servicio que se adquiera. En este caso se desplegó un clúster de HPC haciendo uso del servicio gratuito de AWS.

4.1 Entorno Experimental

Existen varias herramientas para crear clústeres virtuales de manera automática en la plataforma EC2 de Amazon, algunas son StarCluster o CfnCluster. Pero en este caso era necesario desplegar un clúster de forma manual y con una pila de software instrumentada ya que el objetivo del trabajo es un análisis detallado de las diferentes capas que comprenden el sistema.

En la Figura 4 se pueden ver los entornos físicos y virtuales con las respectivas configuraciones.

Se creó un clúster con 5 nodos: un master y 4 nodos de cómputo. La incorporación de PVFS2 obligatoriamente debe ser llevada a cabo antes de instalar MPICH o los benchmarks para que estos programas puedan reconocerlo. Tanto en el clúster físico como en el virtual se utilizó SSH para ejecutar comandos e implementar las configuraciones. Se optó por asignar cuatro nodos de E/S o servidores de datos y un servidor de metadatos o storage. La descripción de las características de los clústeres se puede ver en la Tabla 1.

4.2 Aplicación

Los experimentos fueron diseñados para analizar el comportamiento del sistema frente a la escritura y lectura de tres tamaños de archivos: 1GB, 2GB y 4GB. Las razones de restringir a estos tamaños se deben al objetivo de no prolongar demasiado los tiempos de ejecución y al mismo tiempo no exceder el limitado espacio de almacenamiento ofrecido por el entorno del clúster virtual para almacenar datos. Para conseguirlo

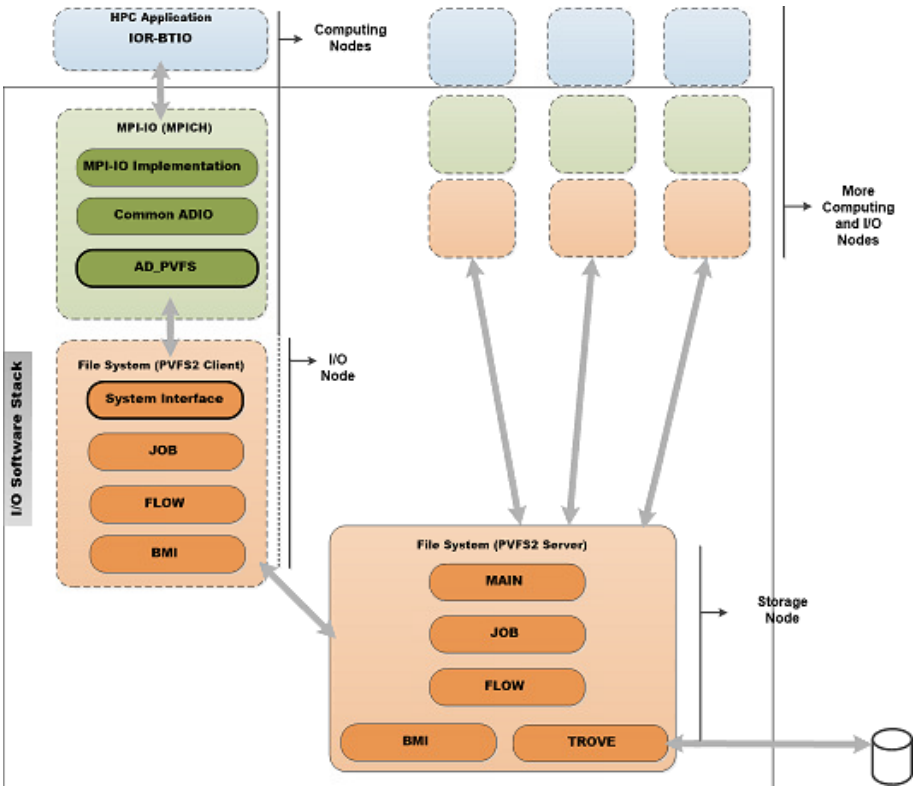

Figure 2: Capas analizadas de la pila de software de E/S

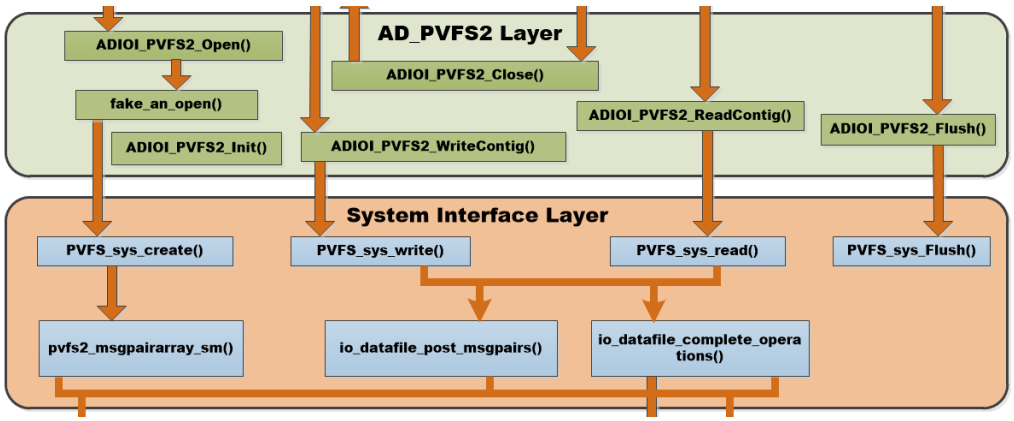

Figure 3: Funciones identificadas en la capa de MPI-IO y PVFS2 del lado del cliente

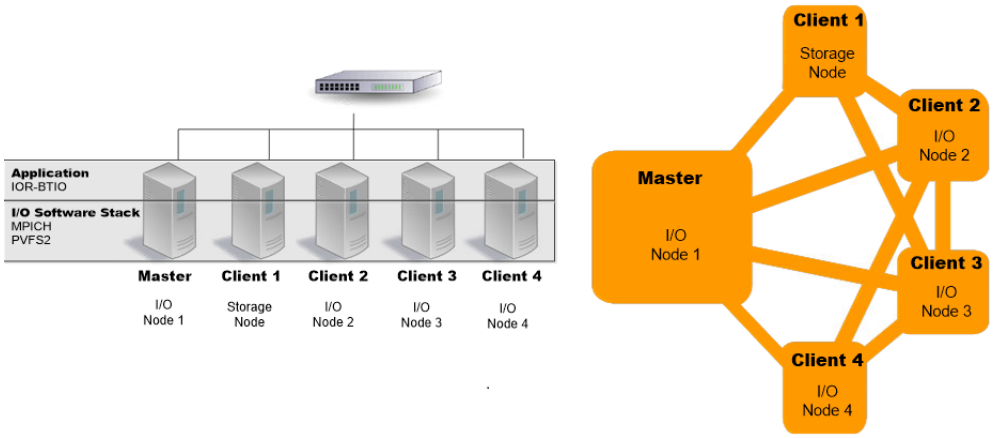

Figure 4: Clusters Físicos (figura en la izquierda) y Virtuales (figura en la derecha)

Table 1: Características de los clústeres utilizados

Componente	Clúster Físico	Clúster Virtual
Procesador	Xeon, i7, AMD	Xeon
RAM (por nodo)	8 GiB	1 GiB
Red de almacenamiento	1 Gbps Ethernet	Baja a Moderada
Número de Data Server	4	4
Número de MetaData Server	1	1
Librería MPI	MPICH versión 3.2.1	MPICH versión 3.2.1
Sistema de Archivos Global	PVFS 2 versión 2.8.2	PVFS 2 versión 2.8.2
Sistema Operativo	Red Hat Enterprise Linux	Red Hat Enterprise Linux

simplemente hubo que utilizar los parámetros adecuados en el caso de IOR. La línea de comandos utilizada fue:

```
mpirun -np NP ./IOR -a $API \
-t $TRANSFER_SIZE -b $BLOCK \
-F -o OUTPUT_FILE
```

En donde:

- `-np` representa el número de procesos MPI
- `-a` establece la interfaz o API de E/S
- `-t` define el tamaño en bytes del buffer de transferencia
- `-F` define que cada proceso trabaja sobre su propio archivo local
- `-b` define el número de bytes contiguos a escribir por proceso
- `-o` se establece el nombre completo del archivo de prueba

En el caso de BTIO se utilizó la clase W pero hubo que modificar el código fuente. Puntualmente se reemplazó la variable niter por el número de iteraciones necesarias para lograr el tamaño de archivo deseado. Ya que la clase seleccionada posee por defecto 200 iteraciones para escribir un archivo de 188 MB. A partir de esta relación se puede determinar fácilmente la cantidad de iteraciones requeridas para el tamaño de archivo buscado. En la Tabla 2 se pueden ver las configuraciones para la utilización de los dos benchmarks.

4.3 Resultados

Para mostrar los resultados se han tomado como ejemplo la capa `AD_PVFS2` que contiene las funciones para implementar la interfaz MPI-IO en PVFS2 y en este caso las funciones `ADIOI_PVFS2_WriteContig()` y `ADIOI_PVFS2_ReadContig()`. También la capa System Interface que es una interfaz del lado cliente que permite la manipulación de los objetos del sistema de archivos y en particular las funciones `PVFS_sys_write()` y `PVFS_sys_read()`. En la

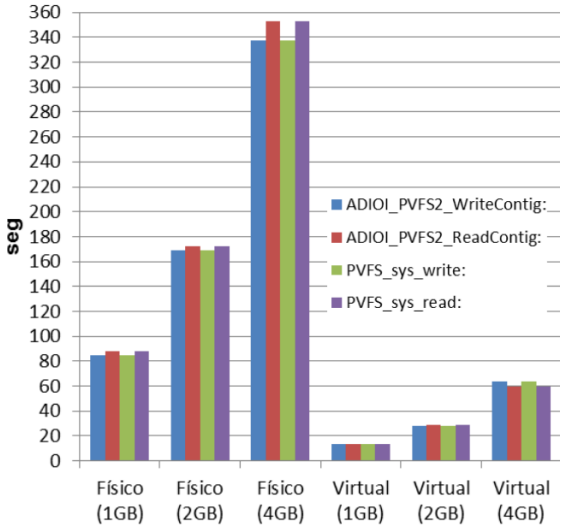

Figure 5: Métricas de IOR

Figura 5 se pueden ver los tiempos obtenidos con el benchmark IOR.

En la Figura 6 se pueden ver los tiempos obtenidos con el benchmark BTIO. Las diferencias con el clúster físico se deben principalmente a las distintas prestaciones de hardware conseguidas en los dos entornos de ejecución pero siguen la misma tendencia.

A pesar que los recursos hardware del clúster físico son ligeramente mejores a los del clúster virtual, en todos los casos el virtual presenta un rendimiento considerablemente superior. Se puede tomar esta situación como confirmación de que la velocidad de red es mucho más inferior en el clúster físico que en el virtual. Sumado al poco peso que poseen los accesos efectivos a disco sobre el tiempo de ejecución total del benchmark, provoca que el incremento en la complejidad de la red supera el beneficio que implica poder realizar múltiples lecturas y escrituras simultáneas en disco en lugar de hacerlas sólo de a una por vez.

Finalmente, se puede observar que con el clúster virtual se dispone de un sistema dedicado con el cual es posible realizar un análisis detallado de la pila de software de E/S. Esto permite definir la importancia temporal de los módulos o funciones de la pila relacionados al flujo de datos en los servidores de E/S

Table 2: Configuraciones seleccionadas para IOR y BTIO

Parámetro	**BTIO**	**IOR**
API para E/S (Interfaz)	MPI-IO	MPI-IO
Tamaño del buffer de transferencia	0,94 MB para 1GB 0,94 MB para 2GB 0,94 MB para 4GB	128 MiB para 1GiB 128 MiB para 2GiB 128 MiB para 4GiB
Tamaño del bloque de datos a escribir por proceso	256 MB para 1 GB 512 MB para 2GB 1 GB para 4GB	256 MiB para 1GiB 512 MiB para 2GiB 1 GiB para 4GiB

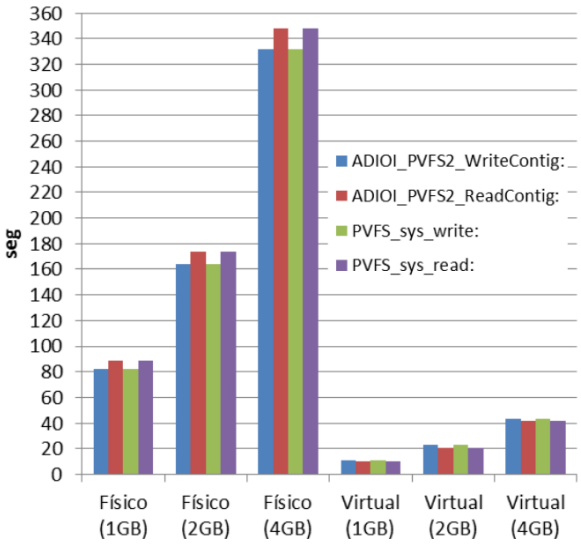

Figure 6: Métricas de BT-IO

como en los servidores de metadatos. La relevancia de conseguir medidas de tiempo en los componentes de la pila es fundamental ya que como se ha explicado anteriormente, distintos patrones de E/S o cargas de trabajo provocarán que el sistema se comporte de diferente manera.

5 Conclusión

El trabajo realizado en Cloud ha permitido identificar las funciones principales en la capa de MPI y del sistema de ficheros paralelo PVFS2 con mayor impacto en el comportamiento del sistema de E/S. Las métricas obtenidas en el clúster virtual no son similares a las encontradas en el clúster físico. Las diferencias se deben principalmente a las distintas prestaciones de hardware conseguidas en los dos entornos de ejecución. Sin embargo, los experimentos demuestran que se puede utilizar el entorno cloud para modificar y medir en el sistema ya que las medidas obtenidas representan el comportamiento de la pila.

Para poder usar esa información y esta técnica en sistemas en producción a nivel de usuario, se está trabajando en la implementación de un trazador (wrapper) para capturar las funciones de interés usando instrumentación dinámica, de manera que no sea necesario recompilar ninguna capa de la pila de software de E/S.

Agradecimientos

This research has been supported by the MICINN/MINECO Spain under contracts TIN2017-84875-P.

References

[1] P. Gomez-Sanchez, D. Encinas, J. Panadero, A. Bezerra, S. Mendez, M. Naiouf, A. D. Giusti, D. Rexachs del Rosario, and E. Luque, "Using AWS EC2 as Test-Bed infrastructure in the I/O system configuration for HPC applications," *Journal of Computer Science & Technology*, vol. Volumen 16, 11/2016 2016.

[2] R. Thakur, W. Gropp, and E. Lusk, "Data Sieving and Collective I/O in ROMIO," in *Proceedings of the The 7th Symposium on the Frontiers of Massively Parallel Computation*, FRONTIERS '99, (Washington, DC, USA), pp. 182–, IEEE Computer Society, 1999.

[3] MPICH, "Mpich overview," 2018.

[4] OrangeFS, "The orangefs project," 2018.

[5] T. PVFS2, "PVFS 2 File System Semantics Document," tech. rep., PVFS Development Team, 2015.