UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA
DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

Máster en Ingeniería de Computadores y Redes

Trabajo Fin de Máster

SISTEMAS DE ALMACENAMIENTO DE DATOS EN CLUSTERS DE COMPUTADORES

Autor: Orlando Landaeta Leal

Directores: Elvira Baydal Cardona - Pedro López Rodríguez

Septiembre 2019

RESUMEN

Desde hace algún tiempo, los clusters de computadores se han popularizado como una alternativa muy interesante desde el punto de vista coste/prestaciones/fiabilidad para la realización de computadores de altas prestaciones, centros de proceso de datos y servidores de Internet. En cualquiera de estos casos, es muy importante que el clúster incorpore un sistema de almacenamiento capaz de dar soporte a los usuarios/clientes del sistema.

En este TFM se analizan diversas alternativas de sistemas de almacenamiento para su empleo en un clúster de computadores. Para cada alternativa considerada, se realiza una descripción somera, su proceso de instalación y una evaluación de su comportamiento.

Palabras clave: instalación, configuración, Unix, virtualbox, almacenamiento, RAID, NFS, GlusterFS, MooseFS, Ceph, DRBD.

INDICE

1. INTRODUCCIÓN

Actualmente, las redes son uno de los medios al que los usuarios recurren para buscar, guardar o compartir información a la que puedan tener acceso de manera remota.

Con el paso del tiempo, cada día es mayor la cantidad de información que usan las personas en casi todas sus actividades, y con el avance de la tecnología también se incrementa la cantidad de información que se almacena en medios electrónicos y se reduce la que se mantiene en papel físicamente.

Cada día son más la cantidad de usuarios y empresas que usan servicios de almacenamiento en la nube como Amazon Web Service, Microsoft Azure y Google Cloud entre las más conocidas aunque también en otros casos implementan su propio sistema de almacenamiento dentro de su red local.

En particular, en entornos de clusters de computadores es frecuente encontrarnos con sistemas que utilizan almacenamiento compartido entre múltiples servidores que acceden al mismo a través de la red. Esto ofrece múltiples ventajas al permitir a los servidores el acceso a los mismos datos. Por ejemplo, facilita el reparto de carga entre servidores que ofrecen el mismo servicio o la tolerancia a fallos, al permitir a un servidor asumir fácilmente, en caso de fallo, los servicios que estaba ofreciendo otro servidor, facilitando de esta forma la administración del clúster.

El acceso al almacenamiento compartido podrá realizarse básicamente siguiendo dos estrategias:

NAS (Network Attached Storage), básicamente acceso servidores de ficheros como NFS o CIFS. En general, con esta estrategia el cliente suele ser consciente de que el almacenamiento es externo y el acceso suele realizarse a través de la red básica.

SAN (Storage Area Network), en este caso se ofrece el acceso al dispositivo de almacenamiento a nivel de bloques de disco y la organización de como se realiza ese acceso reside en el cliente que accede al almacenamiento. En muchos casos se utilizará una red especifica para el acceso a los datos.

La diferencia entre ambas estrategias de almacenamiento, NAS y SAN, puede clarificarse observando la visión que ofrecen al sistema operativo del cliente. Mientras el almacenamiento NAS se ve como un servidor de ficheros (donde el cliente accede a directorios compartidos del servidor como unidades de red), los discos ofrecidos a través de una SAN se muestran al sistema operativo directamente como discos que deben ser formateados con un sistema de ficheros y montados al igual que los locales. En el caso de un clúster, el acceso a la SAN de forma concurrente por varios servidores del clúster podría provocar daños a los datos almacenados en el dispositivo, por lo que será necesario el empleo de sistemas de ficheros de clúster como GlusterFS, OFCS2 o CEPH.

Este proyecto trata sobre la configuración y evaluación de algunas de las diversas opciones de software libre existentes en el mercado y necesarias para ofrecer acceso a sistemas de almacenamiento compartido en ambiente Linux.

Ademas, cualquier dispositivo de almacenamiento siempre existe la posibilidad de que se produzca un fallo, tal es el caso de fallos eléctricos o de hardware, corriendo el riesgo de perder los archivos que se encuentran allí, razón por la cual se hace imprescindible tener redundancia para garantizar la continuidad del acceso a los datos, Otro motivo para hacer copias de seguridad es el ahorro de tiempo al momento en que se necesite traspasar toda la información a otros nodos, por otra parte, es importante también disponer de copias de respaldo que permitan la restauración de los datos en caso de desastres.

Hacer backups ayuda, por tanto, a ahorrar tiempo y a optimizar el proceso de migración de datos cuando cambias de computadora.

En algunos sistemas, las copias de seguridad se crean en dispositivos de almacenamiento instalados dentro del mismo nodo en el que se encuentran los discos con los que se maneja la información principalmente, pero podría ocurrir que el fallo se presente no solo en el disco sino en el nodo a nivel general, es allí donde entra el sistema de almacenamiento DRBD, el cual nos permite trabajar en un nodo mientras la copia seguridad se mantiene actualizada en un nodo diferente.

1.1. Objetivos

El objetivo principal de este proyecto consiste en instalar, configurar y evaluar algunas de las alternativas existentes en sistemas de almacenamiento en clusters usadas actualmente bajo ambiente Linux.

Durante el desarrollo de este trabajo se han implementado las maquinas virtuales necesarias. En cada sistema de almacenamiento varía el numero de máquinas virtuales que se usan dependiendo de como funcionan y se distribuyen las tareas en cada en cada uno de ellos.

Para evaluar cada uno de los sistemas de almacenamiento se realizarán unas pruebas para determinar y comparar la fiabilidad y las prestaciones que ofrece cada uno.

Para evaluar la fiabilidad en cada sistema de almacenamiento se realizarán unas pruebas desconectando algunos de los nodos del clúster y verificando si el sistema continúa funcionando o si por el contrario comienza a presentar fallos.

Para evaluar las prestaciones se realizarán pruebas de lectura y escritura de archivos. Ejecutaremos una aplicación que se encargará de escribir y leer archivos en el

sistema de almacenamiento. Al finalizar el proceso, la aplicación facilitará el tiempo que le ha tomado realizar las pruebas. Los resultados obtenidos en cada uno de los sistemas de almacenamiento serán comparados para verificar los niveles de prestaciones que ofrecen.

2. ESTADO DEL ARTE

Como ya se ha indicado, para el almacenamiento de datos en red básicamente existen principalmente dos tipos de sistemas distribuidos en red, NAS (Network Attached Storage) y SAN (Storage Area Network).

2.1 NAS

Es un único dispositivo de almacenamiento conectado a una red que opera sobre los archivos de datos permitiendo almacenar y recuperar los datos en un punto centralizado al que se accede mediante protocolos de acceso a ficheros como NFS o CIFS.

Tradicionalmente son más económicas que las SAN, suelen ser pequeñas, con menor rendimiento que las SAN y se recomiendan para ciertos servicios.

En muchos casos para ampliar la capacidad y fiabilidad se utilizan varios dispositivos de almacenamiento frecuentemente configurados en RAID (Redundant Arrays of Independent Disks).

2.2 SAN

Es una red local de múltiples dispositivos de almacenamiento conectados en red que operan a nivel de bloques de disco para brindar al usuario almacenamiento unificado,

permitiendo manejar los datos como si fuese un único bloque, usado en sistemas de almacenamiento como GlusterFS y MooseFS.

Tradicionalmente más caras que NAS, suelen usarse para grandes instalaciones, valen para todos los servicios incluyendo grandes servidores de base de datos

2.3 RAID

Un RAID es un grupo de discos independientes configurados para funcionar a nivel lógico como uno solo, ya sea sumando su espacio total, mejorando la velocidad de lectura y escritura o configurados para duplicar la información para estar seguros de que, en caso de que uno de los discos se rompa, no vamos a perder los datos. [1]

Los RAID son muy comunes a la hora de montar un servidor en una empresa o un NAS doméstico. Existen diferentes tipos de RAID, cada uno con unas características propias según la finalidad que busquemos y el número de discos que vayamos a utilizar. [1]

2.3.1 RAID 0

Es uno de los tipos más básicos de RAID, tanto que muchos usuarios ni siquiera lo consideran un tipo RAID como tal. En esta configuración el conjunto de discos funciona como un único volumen, cuyo espacio total es la suma del espacio de todos los discos. [1]

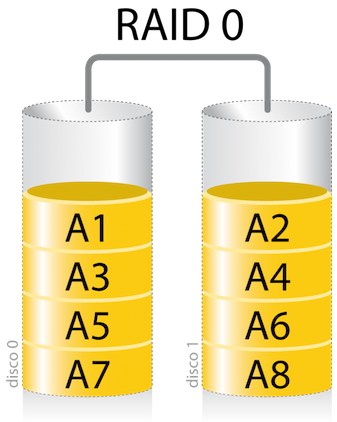

Figura 2.3.1-2.3-1

2.3.2 RAID 1

RAID 1 es uno de los tipos de RAID más utilizados para quienes buscan duplicidad de los datos para estar seguros de que los datos nunca se pierden. En este tipo de RAID, los datos se duplican en todos los discos como si fuesen un espejo. [1]

De esta manera, aunque no tenemos mejora de rendimiento en las velocidades de escritura la velocidad de lectura sí es el doble, dado que los datos se leen a la vez desde las dos unidades. Además, tenemos la seguridad de que si falla uno de los discos, los datos siguen intactos en el segundo y, al reemplazar el dañado, los datos volverán a duplicarse automáticamente. [1]

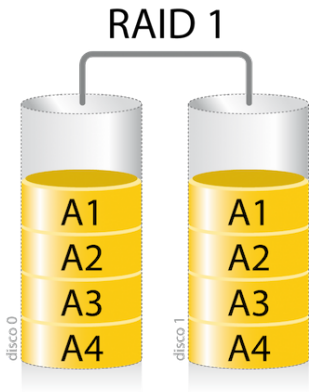

Figura 2.3.2-1

2.3.3 RAID 5

En el RAID 5, la información se distribuye a lo largo de todos los discos duros, aunque se reserva el tamaño de una de las unidades para paridad. Los bloques de paridad, además, se reparte entre todos los discos. [1]

Este tipo de RAID ya es más utilizado en entornos empresariales que en entornos domésticos, aunque si tenemos un NAS con tres o más discos podemos elegirlo para tener una gran ganancia de velocidad de lectura, además de, gracias a la paridad de los datos, poder recuperar toda la información si uno de los discos falla. Si fallan dos perdemos absolutamente toda la información de todo el RAID. [1]

El espacio total de un RAID 5 es el espacio de todos los discos menos uno, es decir, si vamos a usar cuatro discos de 4 TB el espacio total será de 12 TB. La mejora de velocidad de lectura es también X-1 veces el número de discos usados. En el ejemplo anterior, por ejemplo, en el mejor de los casos sería de 3 veces más. [1]

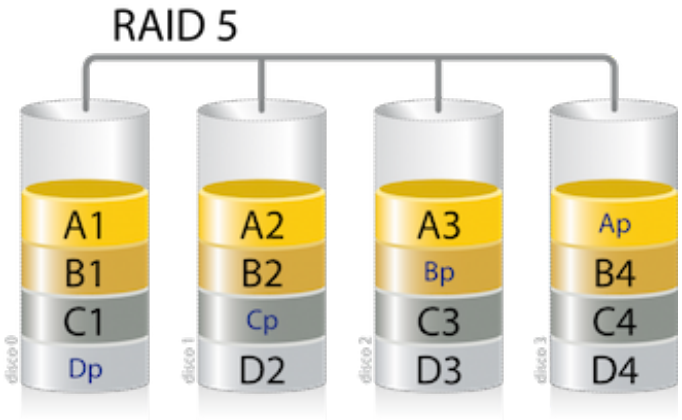

Figura 2.3.3.1

2.3.4 RAID 6

Prácticamente igual que el RAID 5, pero añade un segundo nivel de paridad, lo que nos permite que fallen hasta dos discos del RAID y poder sustituirlos. Si fallan 3, entonces toda la información del RAID se pierde. [1]

A cambio de esta doble paridad incluida en el RAID 6 se pierde el espacio total de dos de los discos. Por ejemplo, en una configuración de 4 discos de 4 TB, el espacio total que tendríamos es de 8 TB, con el doble de velocidad de lectura. [1]

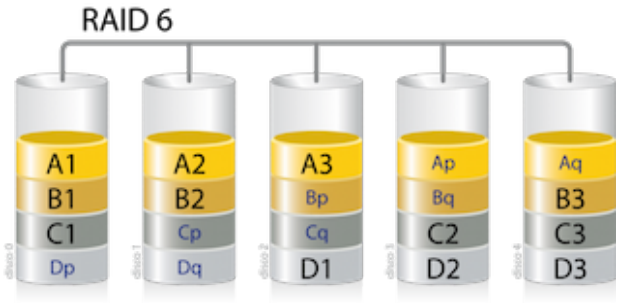

Figura 2.3.4-1

2.3.5 Raid 10 (1+0)

RAID 10 es, a grandes rasgos, un RAID 0 creado a partir de dos configuraciones RAID 1, es decir, creamos dos RAID 1 (espejo) y, a continuación, hacemos un RAID 0 de los dos anteriores. De esta manera tenemos una división de espejos en la que podemos permitirnos que todos los discos, menos uno, puedan fallar en cada RAID 1 sin perder los datos.

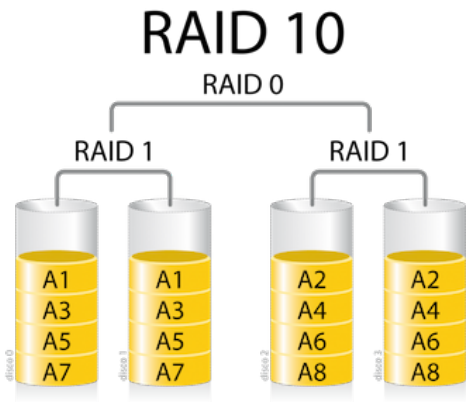

Figura 2.3.5-1

2.4 NFS

NFS (Network File System) es un protocolo de nivel de aplicación, según el modelo OSI, que se utiliza para sistemas de archivos distribuidos en un entorno de red de área local. Permite que diferentes sistemas conectados en una misma red puedan acceder a ficheros remotos como si se tratara de locales. [2]

2.4.1 Ventajas NFS

Entre las principales ventajas podemos encontrar que: [3]

- Se reducen los requerimientos de espacio de disco en las estaciones de trabajo
- La centralización de archivos simplifica las tareas de administración.
- Se puede usar como complemento a NIS.
- Facilita a los usuarios la manipulación de archivos remotos (no necesitan el conocimiento de comandos remotos (ftp, rlogin, rsh, etc.), sino locales (cp, mv, rm, etc.).
- Puede utilizarse para proveer espacio de disco a estaciones que no tengan disco.

- Ayuda a mantener consistencia en los archivos, requisito importante en entornos de clúster.
- Puede utilizarse para la comunicación de usuarios y/o programas.

2.5 GlusterFS

GlusterFS es un sistema de archivos de red paralelo y escalable, adecuado para tareas de uso intensivo de datos como el almacenamiento en la nube. Permite crear soluciones de almacenamiento distribuido por la red y de gran capacidad, además GlusterFS es un software gratuito y de código abierto que puede utilizar hardware común. [4]

2.5.1 Conceptos de GlusterFS

Seguidamente se introducen algunos términos utilizados en un sistema de archivos GlusterFS. [5]

Sistema de archivos distribuido: Es un sistema de archivos en el que los datos se distribuyen entre varios nodos y los usuarios pueden tener acceso a estos datos sin conocer la ubicación real de los archivos. El usuario no experimenta la sensación de acceso remoto.

Brick: Un brick (ladrillo) es básicamente cualquier directorio o partición que será compartido y que está asignado a un volumen.

Volumen: Un volumen es una colección lógica de bricks. Las operaciones se basan en los diferentes tipos de volúmenes creados por el usuario.

2.5.2 Tipos de volúmenes

GlusterFS soporta diferentes tipos de volúmenes adaptándose a los requerimientos. Algunos volúmenes son buenos para escalar el tamaño de almacenamiento, otros para mejorar el rendimiento, y otros para conseguir ambas cosas. [5]

Volumen distribuido: este es el tipo de volumen por defecto en GlusterFS, si no se especifica un tipo de volumen concreto. Los archivos se distribuyen en varios bricks en el volumen, de forma que el archivo 1 sólo podrá almacenarse en el brick 1 o en el brick 2, pero no en ambos, por lo que no habrá redundancia de datos. Este tipo de volumen, al no proporcionar redundancia, puede sufrir la pérdida de los datos en caso de que uno de los dos bricks falle, por lo que es necesario realizar un backup de los archivos con una aplicación externa a GlusterFS.

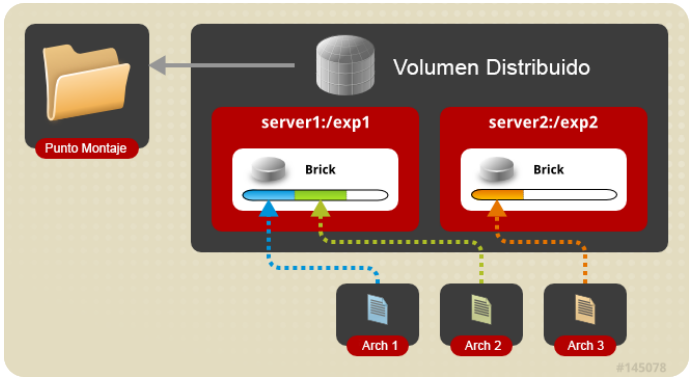

Figura 2.5-1

Volumen replicado: con este tipo de volumen eliminamos el problema ante la pérdida de datos que se experimenta con el volumen distribuido. En el volumen replicado se mantiene una copia exacta de los datos en cada uno de los bricks. El número de réplicas se configura por el usuario al crear el volumen, si queremos dos réplicas necesitaremos al menos dos bricks, Si un brick está dañado, todavía podremos acceder a los datos mediante otro brick. Este tipo de volumen se utiliza para obtener fiabilidad y redundancia de datos.

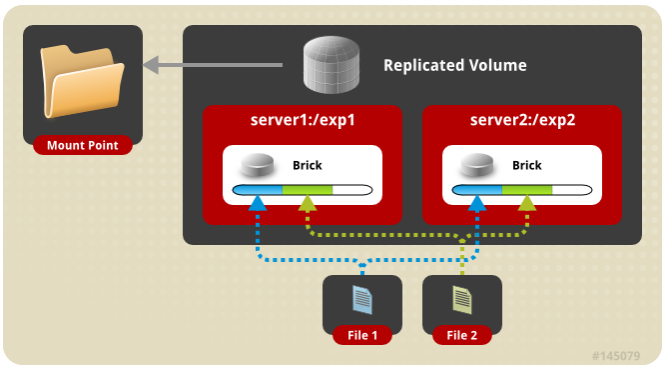

Figura 2.5-2

Volumen distribuido replicado: en este tipo de volumen los datos se distribuyen en conjuntos replicados de bricks. El número de bricks debe ser un múltiplo del número de réplicas. También es importante el orden en que especifiquemos los bricks porque los bricks adyacentes serán réplicas entre ellos. Este tipo de volumen se utiliza cuando se necesita una alta disponibilidad de los datos debido a su redundancia y a la escalabilidad del tamaño de almacenamiento.

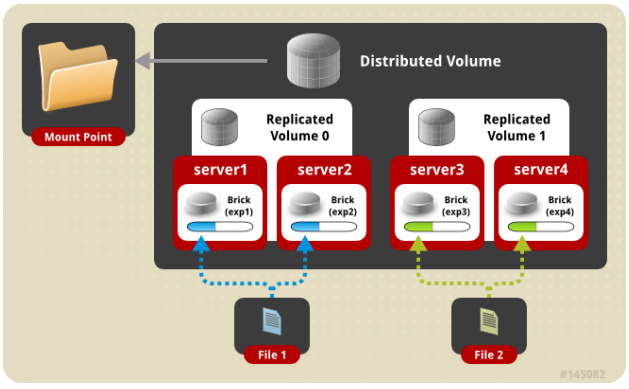

Figura 2.5-3

Volumen seccionado (Striped): en un volumen seccionado los archivos se almacenan en diferentes bricks después de haberse dividido en secciones, de forma que un archivo grande se divide en diferentes secciones y cada sección se almacena en un brick. De este modo se distribuye la carga y el archivo puede ser recuperado más rápidamente, pero en este tipo de volumen perderemos la redundancia de los datos.

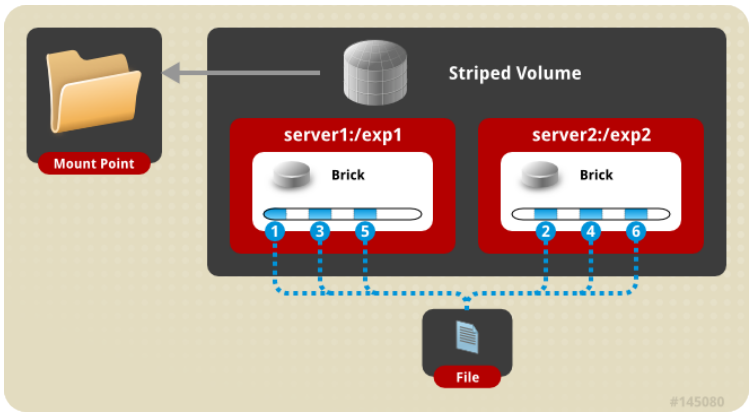

Figura 2.5-4

Volumen seccionado distribuido: este volumen es similar al volumen seccionado, excepto que las secciones en este caso pueden ser distribuidas en más cantidad de bricks.

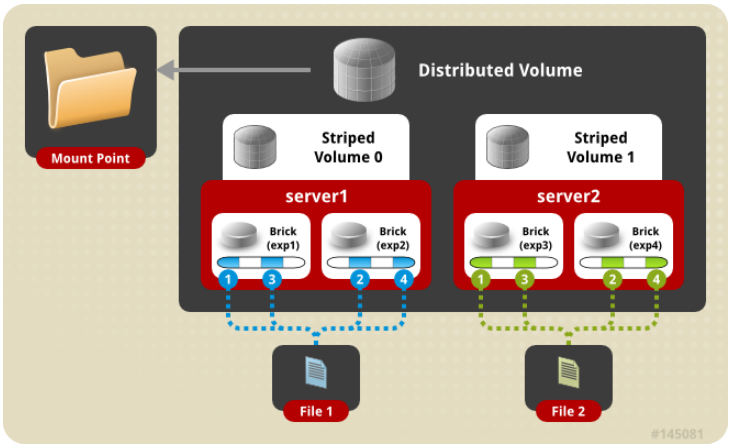

Figura 2.5-5

2.5.3 Ventajas GlusterFS

Como ventajas en GlusterFS encontramos: [6] [7]

- Escala a varios petabytes.
- Maneja miles de clientes.

- Compatible con POSIX.
- Utiliza hardware de productos básicos.
- Puede usar cualquier sistema de archivos que admita atributos extendidos. [6]
- Accesible utilizando protocolos estándar de la industria como NFS y SMB. [6]
- Proporciona replicación, cuotas, geo-replicación, instantáneas y detección de bitrot.
- Permite la optimización de diferentes cargas de trabajo.
- Código abierto.
- No utiliza servidor de meta-datos que lo hace que sea sencillo escalar el servicio en cualquier momento.

2.6 MooseFS

MooseFS es un sistema de archivos distribuido en red, tolerante a fallos, de alta disponibilidad, alto rendimiento y escalamiento horizontal. Difunde los datos en varios servidores físicos básicos, que son visibles para el usuario como un disco virtual. Es compatible con POSIX y actúa como cualquier otro sistema de archivos similar a Unix que admita: [8]

- Estructura jerárquica: archivos y carpetas.
- Atributos del archivo.
- Archivos especiales: tuberías, zócalos, bloques y dispositivos de caracteres.
- Enlaces simbólicos y duros.
- Atributos de seguridad y ACLs.

2.6.1 MooseFS vs MooseFs PRO

El sistema de almacenamiento MooseFS ofrece dos ediciones, la edición MooseFS Pro es de pago, y a diferencia de la edición estándar, incluye soporte con opciones

adicionales como asesores dedicados, asistencia remota, asistencia en el sitio, monitoreo activo pro, lanzamientos de software dedicados. Además, también ofrece características que no están incluidas en la edición estándar como: [8]

- Redundancia de datos con codificación de borrado.
- Alta disponibilidad de metadatos.
- Cliente nativo de Windows.

2.6.2 Arquitectura MooseFS

MooseFS consta de cuatro componentes: [9]

Servidores de administración (servidores maestros): administran todo el sistema de archivos, almacenando metadatos para cada archivo (información sobre el tamaño, los atributos y las ubicaciones de los archivos, incluida toda la información sobre archivos no regulares, es decir, directorios, sockets, tuberías y dispositivos).

Servidores de datos (servidor chunkserver): almacenan los datos de los archivos y los sincronizan entre sí (si se supone que un determinado archivo existe en más de una copia).

Servidores de respaldo de metadatos (servidor metalogger): almacenan registros de cambios de metadatos y descargan periódicamente el archivo principal de metadatos. En MooseFS (no Pro), la máquina de respaldo de metadatos se puede configurar fácilmente como un maestro en caso de un fallo del maestro principal.

Cliente: las computadoras cliente que acceden (montan) los archivos en MooseFS. Los clientes pueden ser cualquier número de máquinas que utilizan el proceso mfsmount para comunicarse con el servidor de administración (para recibir y modificar

los metadatos del archivo) y con los servidores (para intercambiar datos de archivos reales).

Como podemos ver en la figura 2.6.2-1 para leer datos el nodo cliente consulta al maestro en cual de los nodos de almacenamiento se encuentran los archivos para luego solicitárselos al nodo en donde se encuentran.

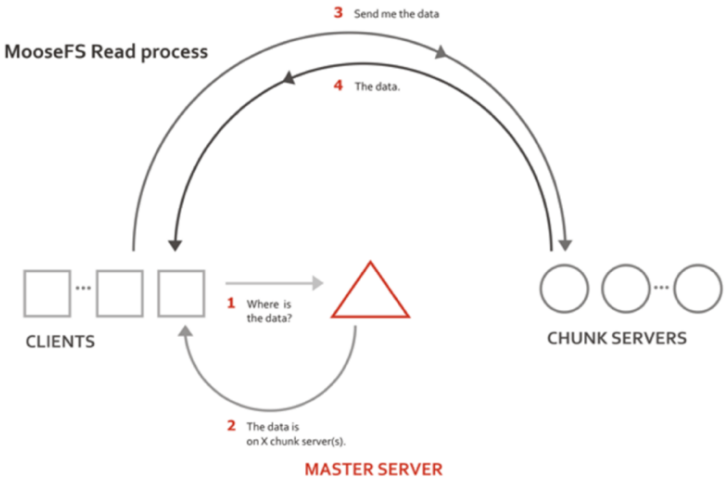

Figura 2.6.2-1

En la figura 2.6.2-2 podemos ver que para escribir datos el servidor maestro le indica al cliente a cuál de los nodos de almacenamiento enviar los archivos. Luego el nodo de almacenamiento se encarga de sincronizarlo con los demás nodos y le confirma al cliente que la información ha sido almacenada.

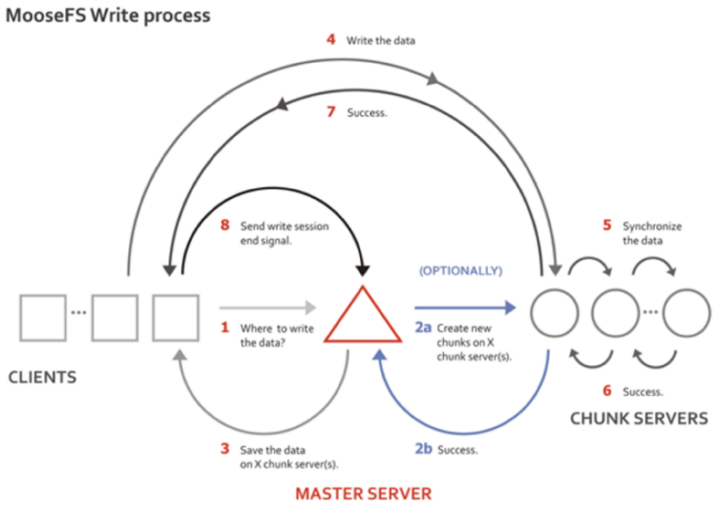

Figura 2.6.2-2

2.6.3 Espacio recomendable en el servidor metalogger

Como se describió anteriormente, el servidor de metadatos se de encarga almacenar los registros de cambios realizados en el servidor maestro. El espacio libre necesario para los metadatos en /var/lib/mfs se puede calcular mediante la siguiente fórmula: [10]

ESPACIO = RAM * (BACK_META_KEEP_PREVIOUS + 2) + 1 * (BACK_LOGS + 1) [GB] (Si se usan los valores predeterminados de /etc/mfs/mfsmaster.cfg, es RAM * 3 + 51 [GB]).

Donde:

- RAM es la cantidad de memoria RAM.
- BACK_LOGS es un número de archivos de registro de cambios de metadatos (el valor predeterminado es 50 desde /etc/mfs/mfsmaster.cfg).
- BACK_META_KEEP_PREVIOUS es un número de archivos de metadatos previos que se conservarán (el valor predeterminado es 1, también de /etc/mfs/mfsmaster.cfg).

El valor 1 (antes de multiplicar por BACK_LOGS + 1) es una estimación del tamaño utilizado por un archivo de registro de cambios [número]. mfs. En instancias altamente cargadas utiliza un poco menos de 1 GB.

Ejemplo:
Si tenemos 128 GiB de RAM en el servidor maestro, utilizando la fórmula dada, debemos reservar para /var/lib/mfs en el servidor maestro: 128 * 3 + 51 = 384 + 51 = 435 GiB mínimo.

Es recomendable configurar una matriz RAID 1 o Raid10 dedicada para almacenar volcados de metadatos y registros de cambios. Dicha matriz se debe montar en el directorio /var/lib/mfs y no debe ser más pequeña que el valor calculado en el punto anterior. No se recomienda almacenar metadatos a través de la red (por ejemplo, SAN, NFS, etc.). [10]

En los servidores de datos de MooseFS no se recomienda el uso de almacenamiento de tipo RAID ya que MooseFS tiene un mecanismo para verificar si el disco está en buenas condiciones o no. Los algoritmos de MooseFS no funcionan con RAID, por lo tanto, las matrices RAID dañadas pueden ser reportadas falsamente como saludables. [10]

2.6.4 Ventajas MooseFS

Entre las ventajas de sistemas de almacenamiento MooseFS encontramos que: [11]

- El almacenamiento definido por software MooseFS está diseñado para aplicaciones de misión crítica con alta disponibilidad y requisitos de alto rendimiento.

- Esta diseñado para soportar operaciones de E/S de alto rendimiento. Los datos del usuario pueden leerse / escribirse simultáneamente en muchos nodos de almacenamiento, evitando así los cuellos de botella de un solo servidor central o de una sola red.

- El software de almacenamiento MooseFS está diseñado para soportar Big Data. MooseFS permite un espacio de almacenamiento virtualmente ilimitado para soportar las cargas de trabajo distribuidas más exigentes.

- El almacenamiento MooseFS se puede construir utilizando componentes básicos x86-64, disponibles de casi todos los fabricantes de hardware. También funciona perfectamente bien con todos los principales discos y tipos de interfaz de disco: SATA / SAS, SSD / HDD.

- El software de código abierto con componentes básicos y sistemas operativos, hace de MooseFS una opción segura y utilizable durante toda la vida.

- Las funciones de actualización por niveles y en movimiento, hacen que el hardware sea más duradero. Los servidores y / o discos más antiguos, más pequeños y más lentos se pueden usar en niveles menos intensivos.

- El almacenamiento se puede ampliar hasta 16 exabytes (~ 16000 petabytes), lo que nos permite almacenar más de 2 mil millones de archivos.

2.7 CEPH

Ceph File System es un sistema de archivos distribuido libre, diseñado para el uso con gran cantidad de datos y muy enfocado para el uso con Big Data. Ceph tiene como objetivo ser POSIX-compatible y completamente distribuido sin ningún punto de fallo. Los datos tienen replicación libre de errores, haciéndolo tolerante a fallos. [12]

2.7.1 Arquitectura CEPH

Los sistemas de archivos Ceph están conformados principalmente por 4 nodos: [13]

Nodo monitor: un monitor Ceph (ceph-mon) mantiene mapas del estado del clúster, incluidos el mapa del monitor, el mapa del administrador, el mapa OSD (Object Storage Daemon) y el mapa CRUSH (Controlled Replication Under Scalable Hashing), un algoritmo que Ceph utiliza para calcular las ubicaciones de almacenamiento de objetos. Estos mapas son un estado de cluster requerido para que los demonios Ceph se coordinen entre sí. Los monitores también son responsables de administrar la autenticación entre demonios y clientes.

Nodo administrador: Un demonio Ceph Manager (ceph-mgr) es responsable de realizar un seguimiento de las métricas de tiempo de ejecución y del estado actual del clúster de Ceph, incluida la utilización del almacenamiento, las métricas de rendimiento actuales y la carga del sistema. Los demonios de Ceph Manager también alojan módulos basados en Python para administrar y exponer la información del clúster de Ceph, incluido un Ceph Dashboard basado en la web y la API REST.

Nodo de almacenamiento: un OSD de Ceph (demonio de almacenamiento de objetos, ceph-osd) almacena datos, maneja la replicación de datos, la recuperación, el rebalanceo y proporciona información de monitoreo a los monitores y administradores de Ceph al verificar otros Demonios de OSD Ceph. Normalmente se requieren al menos 3 Ceph OSD para redundancia y alta disponibilidad.

Nodo de metadatos: Ceph MDS (metadata server) almacena metadatos en nombre del sistema de archivos Ceph (es decir, los dispositivos de bloque Ceph y el almacenamiento de objetos Ceph no usan MDS). Los servidores de metadatos Ceph

permiten a los usuarios del sistema de archivos POSIX ejecutar comandos básicos (como ls, find, etc.) sin poner una carga enorme en el Ceph Storage Cluster.

2.7.2 Copias de seguridad en CEPH

La solución para la recuperación de desastres y la copia de seguridad general a menudo puede ser bastante simplista. Las instantáneas incrementales pueden proporcionar esta y otras funciones bastante bien. [14]

Una instantánea es una copia de solo lectura del estado de una imagen en un momento determinado. Una de las características avanzadas de los dispositivos de bloque Ceph es que puede crear instantáneas de las imágenes para conservar un historial del estado de una imagen. Ceph también admite la creación de capas de instantáneas, lo que le permite clonar imágenes (por ejemplo, una imagen de máquina virtual) de manera rápida y sencilla. Ceph es compatible con las instantáneas de dispositivos de bloque que utilizan el comando rbd y muchas interfaces de nivel superior, incluidas QEMU, libvirt, OpenStack y CloudStack. [15]

2.7.3 Ventajas de CEPH

Como sistema de almacenamiento Ceph ofrece las siguientes ventajas: [16]

- Rendimiento bastante alto.
- Coste relativamente bajo y que se reduce cada vez más según la capacidad aumenta.
- Total independencia de fabricantes.
- Para crecer, sólo es necesario añadir más equipos, crecimiento horizontal.
- Soluciones altamente flexibles.
- Control total sobre la solución.

2.8 DRBD

DRBD es un sistema de almacenamiento replicado distribuido para la plataforma Linux. Se implementa como un controlador de kernel, varias aplicaciones de administración de espacio de usuario y algunos scripts de shell. DRBD se usa tradicionalmente en grupos de computadoras de alta disponibilidad. [17]

DRBD permite réplica de los datos de una partición en varias máquinas. Es decir, teniendo una partición del mismo tamaño en dos máquinas, con DRBD se puede hacer una réplica del contenido de esta partición de forma automática, para que en el caso de que una máquina falle, tenga todo el contenido de esa partición accesible desde la otra máquina. Es como un RAID 1, pero entre distintas máquinas. [18]

Habitualmente, la partición que se replica, solamente está montada en una de las máquinas porque se utiliza un sistema de ficheros tradicional: ext4, xfs, etc. De esta forma, solo una de las máquinas puede acceder a los datos, la que tiene la partición montada. Sirve para montar un sistema de archivos en modo activo/pasivo, y que una de las máquinas tenga todos los datos hasta que falle, y en ese momento se puede acceder desde la otra máquina. [18]

2.8.1 ¿Cómo funciona?

DRDB crea un dispositivo de bloques drbd0 accesible desde ambos servidores. El servidor primario es el que tiene acceso de lectura y escritura en el dispositivo drbd0. Cada vez que escribe algo en drbd0, los datos se escriben en la partición física y esos mismos datos se envían por TCP/IP al servidor secundario que tiene acceso solo de lectura, consiguiendo que ambas particiones físicas estén sincronizadas, exactamente igual que un RAID 1. [19]

2.8.2 Ventajas DRBD

Entre las principales ventajas podemos encontrar que: [19]

- No hay problemas con los recursos del clúster se replica en lugar de compartir.
- Los recursos compartidos de almacenamiento son particularmente sensibles. Esto puede conducir a resultados potencialmente desastrosos en ambos nodos.
- El almacenamiento compartido suele ser caro, consume más espacio y energía. DRBD permite crear una configuración de alta disponibilidad con sólo 2 máquinas.

2.9 Network bonding

Como se ha visto casi todas las alternativas presentadas hacen uso de la red local para intercambiar datos o archivos. Por ese motivo es muy importante, asegurar que un fallo en la red no comprometa todo el sistema.

Network bonding es un método para combinar (unir) dos o más interfaces de red en una sola interfaz. Permite aumentar el rendimiento de la red, el ancho de banda y dará redundancia. Si una interfaz está inactiva o desenchufada, la otra mantendrá el tráfico de la red activo y vivo. La conexión de red se puede utilizar en situaciones en las que necesite redundancia, tolerancia a fallos o redes con equilibrio de carga. [20]

Unix nos permite unir varias interfaces de red en una sola interfaz utilizando un módulo especial del kernel denominado bonding. El bonding de Unix proporciona un método para combinar múltiples interfaces de red en una única interfaz lógica “enlazada”. El comportamiento de las interfaces enlazadas depende del modo; en términos generales, los modos proporcionan de equilibrio de carga. Adicionalmente, se puede realizar monitoreo de integridad del enlace. [20]

2.9.1 Tipos de bonding

Una interfaz de red bonding se compone de varias interfaces de red físicas denominadas esclavos. La interfaz combinada bonding puede funcionar en los siguientes modos: [20]

Política de round-robin: (modo = 0) es el modo por defecto. Transmite paquetes en orden secuencial desde el primer esclavo disponible hasta el último. Este modo proporciona equilibrio de carga y tolerancia a fallos.

Política de copia de seguridad activa: (modo = 1) en este modo, solo un esclavo en el enlace está activo. El otro se activará, solo cuando el esclavo activo falle. La dirección MAC del enlace es visible externamente en un solo puerto (adaptador de red) para evitar confundir al conmutador. Este modo proporciona tolerancia a fallos.

Política XOR: (modo = 2) transmisión basada en [(dirección MAC de origen XOR'd con dirección MAC de destino) módulo recuento de esclavos]. Esto selecciona el mismo esclavo para cada dirección MAC de destino. Este modo proporciona equilibrio de carga y tolerancia a fallos.

Política de difusión: (modo = 3) transmite todo en todas las interfaces esclavas. Este modo proporciona tolerancia a fallos.

Agregación dinámica de enlaces: (modo = 4) crea grupos de agregación que comparten la misma velocidad y configuraciones de dúplex. Utiliza todos los esclavos en el agregado activo de acuerdo con la especificación 802.3ad.

Equilibrio de carga de transmisión adaptativo: (modo = 5) enlace de canales que no requiere ningún soporte de conmutador especial. El tráfico saliente se distribuye de acuerdo con la carga actual (calculada en relación con la velocidad) en cada

esclavo. El tráfico entrante es recibido por el esclavo actual. Si el esclavo receptor falla, otro esclavo asume la dirección MAC del esclavo receptor fallido.

3. ANALISIS DE LOS SISTEMAS DE ALMACENAMIENTO

Para el desarrollo de este trabajo se han realizado pruebas de los diferentes sistemas de almacenamiento como GlusterFS, MooseFS, DRBD y Ceph para comprobar la facilidad de instalación, estabilidad y funcionamiento de cada uno.

En la figura 3-1 se muestra la configuración utilizada para realizar el análisis de los sistemas de almacenamiento. Todos ellos están conectados y pueden accederse desde un nodo cliente, el cual realizara operaciones sobre el sistema de almacenamiento. En una configuración real, el nodo cliente se sustituiría por el clúster de computadores al que se va a asociar el sistema de almacenamiento.

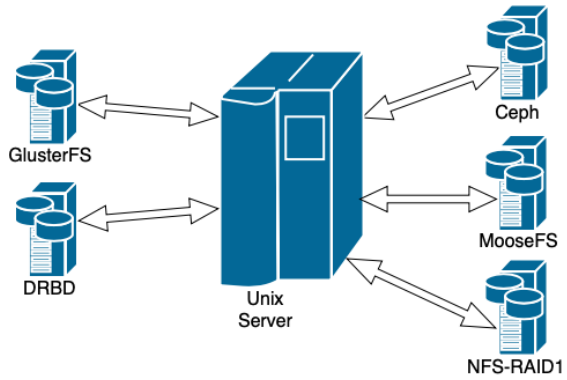

Figura 3-1

El nodo llamado Unix Server en la figura 3-1 el nodo Master en la configuración de cada uno de los sistemas de almacenamiento, este nodo se encargará de proporcionar conexión a internet y facilitar la administración de los nodos en cada sistema de almacenamiento.

3.1 Banco de pruebas

Las pruebas realizadas en cada uno de los sistemas de almacenamiento fueron a través de maquinas virtuales usando la aplicación VirtualBox de Oracle, donde cada uno de los nodos tendrá la misma configuración. Las especificaciones que se asignaron a los nodos fueron las siguientes:

- CPU Intel Core 3.5 GHz de un núcleo.
- Memoria RAM 512 MB.
- Disco duro 5GB para la instalación del sistema operativo.
- Discos duros 5GB cada uno en las unidades de almacenamiento.
- Sistema Operativo Unix Ubuntu 14.04 64 bit.
- Dos tarjetas de red.

Las tarjetas de red de cada nodo en los sistemas de almacenamiento serán configuradas con network bonding utilizando la política de round-robin (modo 0) para obtener equilibrio de carga y tolerancia a fallos. De esta manera, si una de las tarjetas de red deja de funcionar, la conexión a la red se mantendrá conectada a la red con la segunda tarjeta como se muestra en la imagen 3.1.-1.

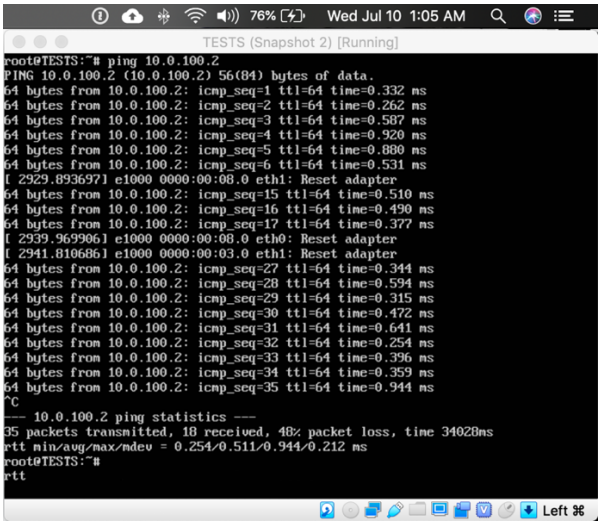

Figura 3.1-1

Las pruebas de lectura y escritura de datos en cada uno de los sistemas de almacenamiento fueron realizadas con la herramienta FIO (Flexible I/O Tester), la cual es una herramienta de código abierto que se utiliza habitualmente para la verificar el rendimiento como prueba de referencia de las unidades HDD, SSD y PCIe.

Para realizar las pruebas con FIO se usarán los siguientes parametros:

Name: es el nombre de los ficheros que se generaran durante los tests.

Ioengine: Define cómo el trabajo emite E/S. en estas pruebas se usará libaio que es I / O asíncrono nativo de Linux.

Iodepth: Número de unidades de E/S para mantener en vuelo contra el archivo. Tenga en cuenta que el aumento de iodepth más allá de 1 no afectará a los motores sincrónicos Incluso los motores asíncronos pueden imponer restricciones de sistema operativo que impiden alcanzar la profundidad deseada. Esto puede suceder en Linux cuando se usa libaio y no se establece directamente iodepth = 1, ya que la E/S almacenada en búfer no es asíncrona en ese sistema operativo.

Rw: tipo de patrón de E/S. Los valores a usar en las pruebas son: read (lectura secuencial), write (escritura secuencial), randread (lectura aleatoria), randwrite (escritura aleatoria) y readwrite (mezcla de lectura y escritura secuencial).

Bs: tamaño de bloque para unidades de E/S. Predeterminado: 4k. Los valores para lecturas y escrituras se pueden especificar por separado en el formato de lectura, escritura, cualquiera de los cuales puede estar vacío para dejar ese valor en su valor predeterminado.

Las pruebas se pueden realizar con bloques pequeños o grandes de E/S, de 512 bytes, 4K, 16K, 32K 64K, 1MB, para obtener la medición adecuada que se puede

visualizar como un histograma. Esto hace que sea más fácil de interpretar. En las pruebas a realizar los bloques serán de 32k.

Size: el tamaño de los archivos generados, las pruebas se realizarán con archivos de 1000 MB.

Numjobs: es el numero de trabajos en paralelo, las pruebas de este documento se realizarán con uno y dos trabajos.

Runtime: limita el tiempo de ejecución en segundos de la prueba.

Rwmixread: Porcentaje de una carga de trabajo mixta que debe leer y escribir, cuando el parámetro --rw utiliza lecturas y escrituras la regla general es 75/25, aunque realmente depende de las cargas de trabajo, la regla general es que hay un 75% de lecturas y un 25% de escrituras en el conjunto de datos, las pruebas se realizaran asignándole un valor de 75.

Debemos tomar en cuenta en cuenta, que un servidor virtual nunca dará el mismo rendimiento que un servidor físico y la velocidad de lectura y escritura en los discos puede variar según las características de fabricación por lo que debemos tener presente que los resultados de estas pruebas podrían cambiar según el hardware del que dispongamos.

3.2 Instalación y pruebas de NFS

Como se muestra en la figura 3.2-1 para la instalación y configuración del servicio solo hemos usado un nodo con dos discos duros. En esta prueba, el tipo de almacenamiento será un sistema RAID 1 implementado en software. Adicionalmente se realizo la misma instalación y configuración con el tipo de almacenamiento RAID 5 usando 3 disco duros.

Figura 3.2-1

3.2.1 Instalación de RAID 1

- En el nodo NFSRAID1

  Instalación del paquete mdadm.

```
apt install mdadm
```

  Se crea el disco RAID

```
# mdadm --create --verbose /dev/md0 --level=1 --raid-
devices=2 /dev/sdb1 /dev/sdc1
```

  Se formatea y se monta el disco.

```
mkfs.ext4 /dev/md0
mkdir /mnt/raid
mount /dev/md0 /mnt/raid
```

  - Se modifica el archivo `/etc/fstab` para que el disco se monte de forma automática al reiniciar el nodo.

```
/dev/md0      /mnt/raid  ext4  defaults   0     0
```

  - Se modifica el archivo `/etc/exports` para exportar el directorio.

```
/mnt/raid 10.0.100.0/24(rw,no_root_squash,sync,no_subtree_check)
```

3.2.2 Instalación de NFS para el RAID 1

- En el nodo NFSRAID1Client

  - Se instala el cliente nfs:

    ```
    apt install nfs-common
    ```

  - Creamos el directorio desde donde accederemos al disco:

    ```
    mkdir /mnt/raid
    ```

  - Se realiza el montaje del disco:

    ```
    mount -t nfs nfaraid1:/mnt/raid /mnt/raid
    ```

  - Para no perder el montaje del disco cuando se reinicie el nodo, modificamos el archivo /etc/fstab agregando la siguiente orden:

    ```
    nfsraid:mnt/raid /mnt/raid nfs auto,_netdev 0 0
    ```

3.2.3 Dificultades encontradas en RAID 1

Durante la instalación del RAID1 no se presentó ninguna dificultad. El único hecho inesperado fue al reiniciar el nodo luego de finalizar la instalación, se presentaba un mensaje de error indicando que la unidad /dev/md0 no existía, por lo que no se podía montar la unidad RAID. Esto se resolvió modificando el archivo /etc/fstab ya que al revisar la lista de discos notamos que el nombre había cambiado. Alternativamente, se puede actualizar el archivo de inicio para que almacene la configuración correcta durante el arranque.

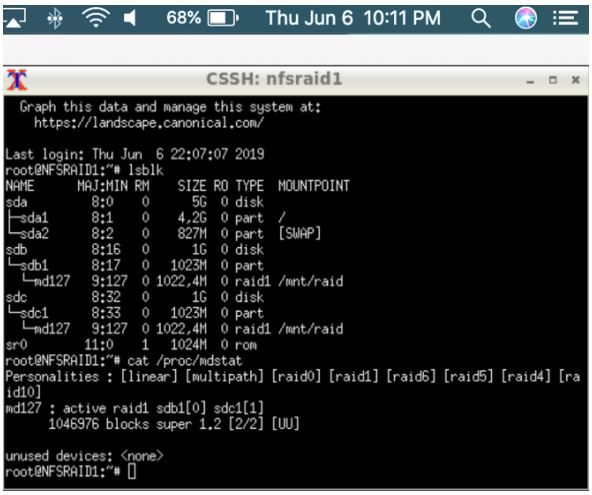

Figura 3.2.3-1

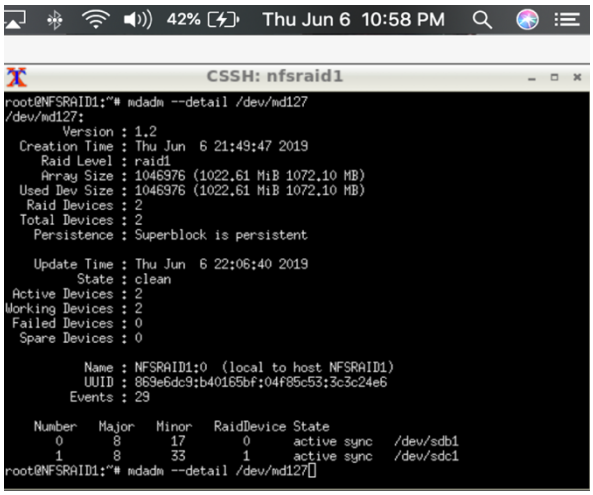

Figura 3.2.3-2

3.2.4 Pruebas en RAID 1

Al realizar la prueba de lectura, lectura aleatoria, escritura, escritura aleatoria y, lectura y escritura simultaneas con la aplicación FIO y realizar una carga de archivos de 1 GB en RAID 1 se presentaron los siguientes resultados:

	1 thread		2 threads	
Read	52971 KB/s		56123 KB/s	
Rand read	32707 KB/s		41559 KB/s	
Write	17871 KB/s		20902 KB/s	
Rand write	13198 KB/s		14856 KB/s	
Read / write	18056 KB/s	7791 KB/s	20824 KB/s	8985 KB/s

Otra de las pruebas realizadas en RAID 1 fue desconectando los discos de la configuración RAID. Al desconectar un disco, el acceso a los datos se mantiene gracias al segundo perteneciente al mismo RAID 1.

3.2.5 Instalación de RAID 5

- En el nodo NFSRAID5

  Instalación del paquete mdadm.

  ```
  apt install mdadm
  ```

  Se crea el disco md0

  ```
  # mdadm --create --verbose /dev/md0 --level=5 --raid-
  devices=3 /dev/sdb1 /dev/sdc1 /dev/sdd1
  ```

  Se formatea y se monta el disco.

  ```
  mkfs.ext4 /dev/md0
  mkdir /mnt/raid
  mount /dev/md0 /mnt/raid
  ```

  - Se modifica el archivo `/etc/fstab` para que el disco se monte de forma automática al reiniciar el nodo.

    ```
    /dev/md0      /mnt/raid  ext4  defaults   0     0
    ```

  - Se modifica el archivo `/etc/exports` para exportar el directorio.

    ```
    /mnt/raid 10.0.100.0/24(rw,no_root_squash,sync,no_subtree_check)
    ```

3.2.6 Instalación de NFS para RAID 5

- En el nodo NFSRAID5Client

  - Se instala el cliente nfs:

    ```
    apt install nfs-common
    ```

  - Creamos el directorio desde donde accederemos al disco:

    ```
    mkdir /mnt/raid
    ```

- Se realiza el montaje del disco:

```
mount -t nfs nfaraid5:/mnt/raid /mnt/raid
```

- Para no perder el montaje del disco al cuando se reinicie el nodo, modificamos el archivo /etc/fstab agregando la siguiente orden:

```
nfsraid:mnt/raid /mnt/raid      nfs     auto,_netdev 0      0
```

3.2.7 Pruebas en RAID 5

Al realizar la prueba de lectura, lectura aleatoria, escritura, escritura aleatoria y, lectura y escritura simultaneas con la aplicación FIO y realizar una carga de archivos de 1 GB en RAID 5 se presentaron los siguientes resultados:

	1 thread		2 threads	
Read	54821 KB/s		58925 KB/s	
Rand read	33416 KB/s		41922 KB/s	
Write	13541 KB/s		17172 KB/s	
Rand write	8071 KB/s		10283 KB/s	
Read / write	13635 KB/s	5883 KB/s	16986 KB/s	7329 KB/s

En caso del sistema de archivos RAID 5, nos permite tener algún fallo en uno de los discos y seguir teniendo acceso a la información almacenada en el RAID 5 sin perder datos. Al asignar un nuevo disco al RAID, el RAID 5 reconstruye los datos a partir de los discos restantes, mientras que si se presentan fallos en más de un disco en el RAID 5 no se podrán recuperar los datos.

3.3 Instalación y Pruebas de GlusterFS

Como se muestra en la figura 3.3-1 para la instalación y configuración del servicio GlusterFS hemos utilizado 4 nodos, 1 nodo cliente (glusterfsclient) y 3 nodos de almacenamiento, que hemos denominado glusterfs01, glusterfs02 y glusterfs03.

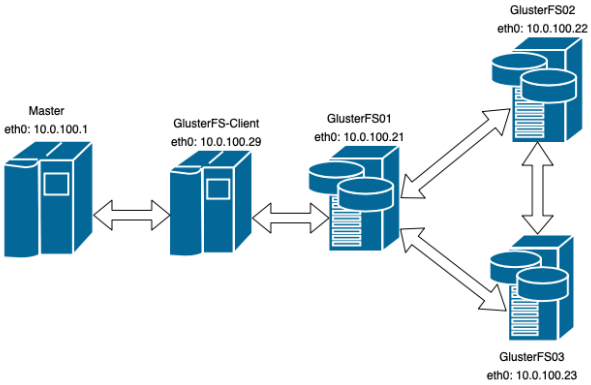

Figura 3.3-1

3.3.1 Instalación de GlusterFS

Realizaremos la instalación de un volumen replicado y otro distribuido.

- Para ello en todos los nodos GlusterFS ejecutaremos los siguientes pasos.

  - Formatear los discos con el sistema de archivos xfs.

```
# mkfs.xfs /dev/sdb1
# mkfs.xfs /dev/sdc1
```

  - Crear los puntos de montaje: volgfs-r para los discos de volumen replicado y volgfs-d para los discos de volumen distribuido.

```
# mkdir –p /volgfs-r
# mkdir –p /volgfs-d
```

  - Realizar el montaje de los discos.

```
# mount /dev/sdb1 /volgfs-r
# mount /dev/sdc1 /volgfs-d
```

  - Crear el subdirectorio que contendrá los datos:

```
# mkdir /volgfs-r/data
# mkdir /volgfs-d/data
```

- Editar el archivo /etc/fstab

```
/dev/sdb1  /volgfs-r  xfs   auto,defaults    0     0
/dev/sdc1  /volgfs-d  xfs   auto,defaults    0     0
```

- A continuación se procede al montaje.

```
# mount -a && mount
```

- Instalar los paquetes necesarios:

```
# apt install glusterfs-server
```

- Añadir clusters al pool [glusterfs02, glusterfs03]

  - El siguiente paso es añadir glusterfs02 y glusterfs03 al pool de almacenamiento para esto ejecutamos en glusterfs01:

```
# gluster peer probe glusterfs02
# gluster peer probe glusterfs03
```

- Configuración de GlusterFS [glusterfs01]

  - Configuraremos el grupo de nodos de almacenamiento, lanzando la siguiente orden sólo desde uno de los nodos que componen el grupo (por ejemplo, desde glusterfs01):

```
# gluster peer status
```

(Si todo ha funcionado correctamente mostrará información sobre los otros dos nodos del grupo, e indicará “State: Peer in cluster (Connected)”).

```
root@glusterfs01:~# gluster peer status
Number of Peers: 3

Hostname: glusterfs03
Uuid: 99f1adfa-3455-4af3-995b-2e80825a4f92
State: Peer in Cluster (Connected)
```

```
Hostname: glusterfs02
Uuid: 0f615380-2c45-4e57-90e2-c895e404792b
State: Peer in Cluster (Connected)
```

- Preparación de un volumen replicado.

```
# gluster volume create gv0 replica 3 glusterfs01:/volgfs-
r/data glusterfs02:/volgfs-r/data  glusterfs03:/volgfs-
r/data
# gluster volume start gv0
# gluster volume info
```

- Preparación de un volumen distribuido.

```
# gluster volume create gv1 glusterfs01:/volgfs-d/data
glusterfs02:/volgfs-d/data    glusterfs03:/volgfs-d/data
# gluster volume start gv1
```

- Montaje y uso de los volúmenes en el nodo [glusterfsclient]:

  - Se instalan los paquetes para el cliente

```
apt install glusterfs-client
```

  - Se crean los directorios respectivos para los volúmenes replicado y distribuido:

```
mkdir -p /mnt/gfs-r
mkdir -p /mnt/gfs-d
mount -t glusterfs glusterfs01:/gv0 /mnt/gfs-r
mount -t glusterfs glusterfs01:/gv1 /mnt/gfs-d
```

  - Modificamos el archivo /etc/fstab para que se monten automáticamente en caso de reiniciar el nodo:

```
glusterfs01:/gv0 /mnt/gfs-r glusterfs defaults,_netdev 0    0
glusterfs01:/gv1 /mnt/gfs-d glusterfs defaults,_netdev 0    0
```

Al finalizar estos pasos hemos conseguido que los nodos funcionen correctamente, como podemos ver en la figura 3.3.1-1, se ha configurado una carpeta de volumen

replicado y otra de volumen distribuido. Se realizaron algunas pruebas de copia de archivos sin que se presentaran dificultades.

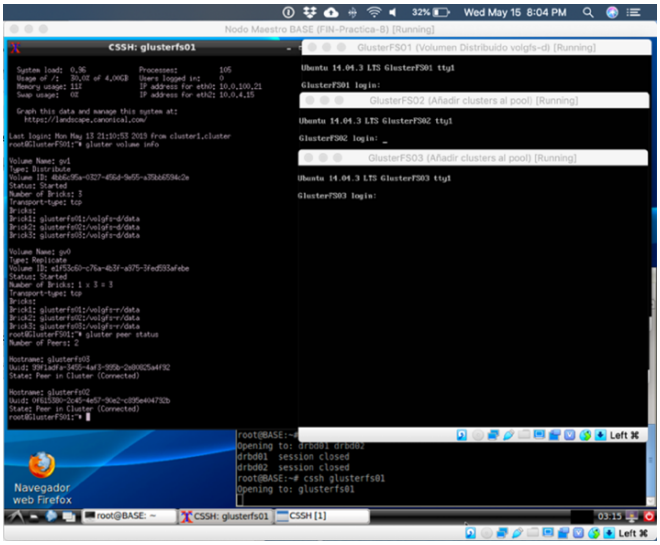

Figura 3.3-2

3.3.2 Pruebas en GlusterFS

Al realizar las pruebas de lectura, lectura aleatoria, escritura, escritura aleatoria y, lectura y escritura simultaneas con la aplicación FIO y realizar una carga de archivos de 1 GB en GlusterFS se presentaron los siguientes resultados:

GlusterFS replicado				
	1 thread		2 threads	
Read	108604 KB/s		135887 KB/s	
Rand read	24992 KB/s		38118 KB/s	
Write	76903 KB/s		84463 KB/s	
Rand write	40519 KB/s		46530 KB/s	
Read / write	20822 KB/s	12783KB/s	29626 KB/s	8984KB/s

GlusterFS distribuido				
	1 thread		2 threads	
Read	111183 KB/s		118067 KB/s	
Rand read	24336 KB/s		25687 KB/s	
Write	149391 KB/s		156093 KB/s	
Rand write	78164 KB/s		79371 KB/s	
Read / write	29792 KB/s	12854KB/s	34590 KB/s	9144KB/s

El sistema de archivos GlusterFS en el volumen replicado nos permite tener algún fallo en uno de los discos o también se podría presentar el caso de que se caiga un nodo y seguir teniendo acceso a la información almacenada, esto posible ya que en GlusterFS los discos que conforman el volumen replicado se encuentran en nodos diferentes.

3.4 Instalación y pruebas de MooseFS

La ejecución de las órdenes para la instalación en cada uno de los nodos y pruebas de funcionamiento se realizaron enviando las órdenes desde un servidor master a través de la aplicación cssh.

Para las pruebas realizadas se procedió a realizar la instalación usando 4 nodos en donde cada uno realiza los procedimientos correspondientes para el correcto funcionamiento del sistema de almacenamiento MooseFS.

Los pasos realizados para la implementación de MooseFS que se detallan mas adelante en el trabajo es las uniones de varias guías de instalación encontradas en distintas paginas web. Esto debido a que se encontraron algunos pasos necesarios que no se especifican la guía publicada en la web oficial de MooseFS, las cuales permitieron conseguir que finalmente el sistema de almacenamiento funcionara correctamente.

Como se muestra en la Figura 3.4-1 para la instalación y configuración del servicio MooseFS hemos utilizado 4 nodos, 1 servidor principal, 1 servidor de almacenamiento y 1 servidor de metadatos y 1 servidor de cliente.

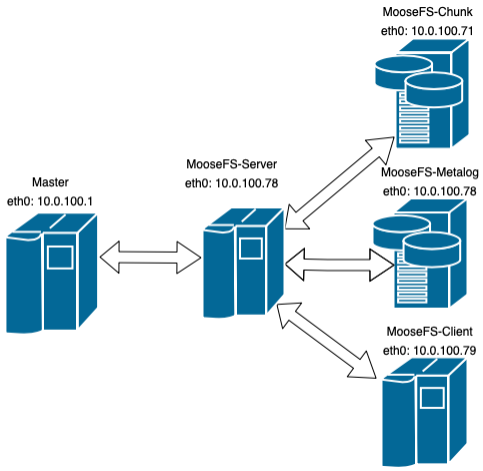

Figura 3.4-1

3.4.1 Instalación de MooseFS

- En todos los servidores:

```
# apt update
# wget -O - https://ppa.moosefs.com/moosefs.key | apt-key
add -
# echo "deb http://ppa.moosefs.com/moosefs-
3/apt/ubuntu/trusty trusty    main" >
/etc/apt/sources.list.d/moosefs.list
```

- En el servidor Master

```
# apt update
# apt install moosefs-master moosefs-cgi moosefs-cgiserv
moosefs-cli
```

  - Editar el archivo /etc/default/moosefs-master

```
MFSMASTER-ENABLE = true
```

  - Editar el archivo /etc/default/moosefs-cgiserv

```
MFSCGISERV_ENABLE = true
```

```
# cd /etc/mfs
# cp mfsmaster.cfg.sample mfsmaster.cfg
# cp mfsexports.cfg.sample mfsexports.cfg
```

- Editar el archivo /etc/mfs/mfsexports.cfg para especificar qué computadoras de los usuarios y los 4 servidores que pueden montar el sistema de archivos, y con qué privilegios.

```
  10.0.100.1/24   /     rw,alldirs,maproot=0
  10.0.100.70/24  /     rw,alldirs,maproot=0
  10.0.100.71/24  /     rw,alldirs,maproot=0
  10.0.100.78/24  /     rw,alldirs,maproot=0
  10.0.100.79/24  /     rw,alldirs,maproot=0
```

- Editar /etc/mfs/mfsmaster.cfg eliminando elcaracter “#” que por defecto en la instalación aparece al inicio de cada línea de las que se muestran a continuación, para permitir que las órdenes se ejecuten:

```
  WORKING_USER = mfs
  WORKING_GROUP = mfs
  SYSLOG_IDENT = mfsmaster
  LOCK_MEMORY = 0
  NICE_LEVEL = -19
  DATA_PATH = /var/lib/mfs
  EXPORTS_FILENAME = /etc/mfs/mfsexports.cfg
  BACK_LOGS = 50

  MATOML_LISTEN_HOST = *
  MATOML_LISTEN_PORT = 9419

  MATOCS_LISTEN_HOST = *
  MATOCS_LISTEN_PORT = 9420

  REPLICATIONS_DELAY_INIT = 300
  REPLICATIONS_DELAY_DISCONNECT = 3600

  CHUNKS_LOOP_TIME = 300
  CHUNKS_DEL_LIMIT = 100
  CHUNKS_WRITE_REP_LIMIT = 1
  CHUNKS_READ_REP_LIMIT = 5
```

```
MATOCU_LISTEN_HOST = *
MATOCU_LISTEN_PORT = 9421

REJECT_OLD_CLIENTS = 0
```

- Ejecutar las siguientes órdenes para iniciar el servicio

```
service moosefs-master start
configuring mfscgiserv autostart
service moosefs-cgiserv start
```

- En el servidor Metalogger:

```
# apt update
# apt install moosefs-metalogger
```

- Editar el archivo /etc/default/moosefs-metalogger

```
MFAMETALOGGER_ENABLE = true
```

```
# cd /etc/mfs
# cp mfsmetalogger.cfg.sample mfsmetalogger.cfg
```

- Editar /etc/mfs/mfsmetalogger.cfg eliminando el caracter “#” al igual como hizo en el nodo mfsmaster:

```
WORKING_USER = mfs
WORKING_GROUP = mfs
SYSLOG_IDENT = mfsmetalogger
LOCK_MEMORY = 0
NICE_LEVEL = -19
DATA_PATH = /var/lib/mfs
BACK_LOGS = 50
META_DOWNLOAD_FREQ = 24

MASTER_RECONNECTION_DELAY = 5
MASTER_HOST = mfsmaster

MASTER_PORT = 9419
MASTER_TIMEOUT = 60
```

- Ejecutamos la siguiente orden

```
# echo "10.0.100.70 mfsmaster" >> /etc/hosts
```

- Iniciamos el servicio

```
# service moosefs-metalogger start
```

- En el servidor Chunkserver:

```
# apt install moosefs-chunkserver
# apt update
```

- Editar el archivo /etc/default/moosefs-chunkogger

```
MFSCHUNKSERVER_ENABLE = true
```

- Preparamos los archivos de configuración

```
# cd /etc/mfs
# cp mfschunkserver.cfg.sample mfschunkserver.cfg
# cp mfshdd.cfg.sample mfshdd.cfg
```

- Editar /etc/mfs/mfschunkserver.cfg eliminando el caracter “#” igual como hizo en el nodo mfsmaster:

```
WORKING_USER = mfs
WORKING_GROUP = mfs
SYSLOG_IDENT = mfschunkserver
LOCK_MEMORY = 0
NICE_LEVEL = -19
DATA_PATH = /var/lib/mfs

BIND_HOST = *
MASTER_HOST = mfsmaster
MASTER_PORT = 9420
MASTER_TIMEOUT = 60
MASTER_RECONNECTION_DELAY = 5

CSSERV_LISTEN_HOST = *
CSSERV_LISTEN_PORT = 9422
CSSERV_TIMEOUT = 5
```

- Se monta el disco

```
mkfs.ext4 /dev/sdb1
mkdir /mnt/mfs1
mount /dev/sdb1 /mnt/mfs1
```

- Editar /etc/mfs/mfshdd.cfg para indicarle los discos a usar

```
/mnt/mfs1
```

- Agregamos el disco al archivo /etc/fstab:

```
/dev/sdb1     /mnt/mfs1  ext4  defaults   0     0
```

- Ejecutamos las siguiente órdenes

```
# chown -R mfs:mfs /mnt/mfs1
# echo "10.0.100.70 mfsmaster" >> /etc/hosts
```

- En el servidor Cliente:

```
# apt install moosefs-client
# apt-get update
# apt install fuse-devel
# apt install fuse-utils
```

- Para habilitar el montaje automático del cliente MooseFS durante el arranque, primero se instalan los paquetes de fuse-devel y fuse-libs.

```
# mkdir -p /mnt/mfs1
# mfsmount /mnt/mfs1 -H 10.0.100.70
```

- Se agrega la siguiente entrada en el archivo /etc/fstab

```
mfsmaster=mfsmaster,mfsport=9421,_netdev 0 0
```

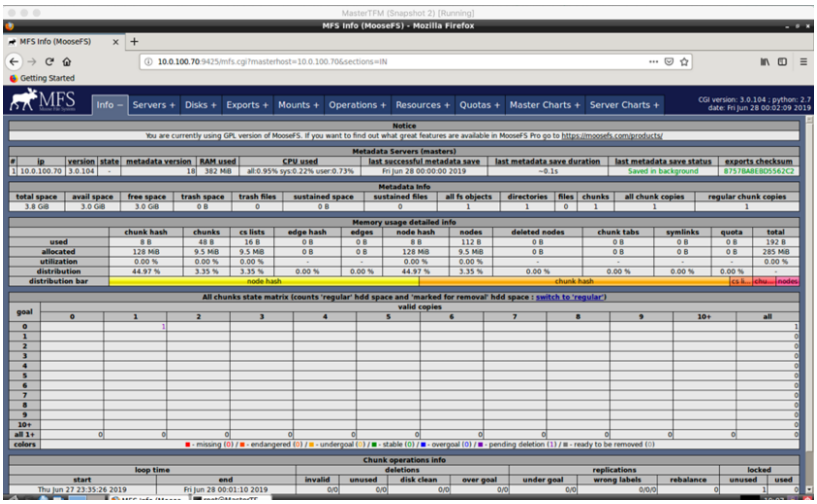

Figura 3.4.1-1

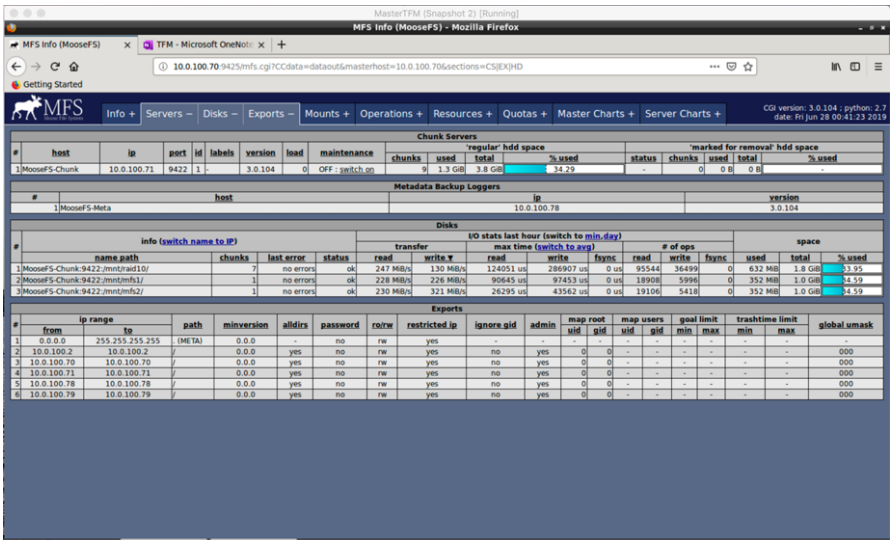

Figura 3.4.1-2

Al finalizar la instalación, configuración y realizadas algunas pruebas hemos podido darnos cuenta de que MooseFS es un sistema de almacenamiento rápido y estable, también cuenta con una aplicación CGI donde muestra información del estado del sistema de almacenamiento como se puede apreciar en las figuras 3.4.1-1 y 3.4.1-2.

3.4.2 Pruebas en MooseFS

Al realizar la prueba de lectura, lectura aleatoria, escritura, escritura aleatoria y, lectura y escritura simultaneas con la aplicación FIO y realizar una carga de archivos de 1 GB en MooseFS se presentaron los siguientes resultados:

	1 thread		2 threads	
Read	134536 KB/s		177769 KB/s	
Rand read	129166 KB/s		144981 KB/s	
Write	39568 KB/s		59008 KB/s	
Rand write	39553 KB/s		54798 KB/s	
Read / write	77571 KB/s	33470 KB/s	75492 KB/s	32556 KB/s

3.5 Instalación y Pruebas de Ceph

Para la instalación y configuración de Ceph se usaron 4 nodos, el nodo ceph01 como servidor, los nodos ceph02 y ceph03 para el almacenamiento y el nodo ceph04 como cliente. Durante el proceso de instalación y configuración se usó un nodo maestro desde donde se ejecutaban las ordenes via ssh.

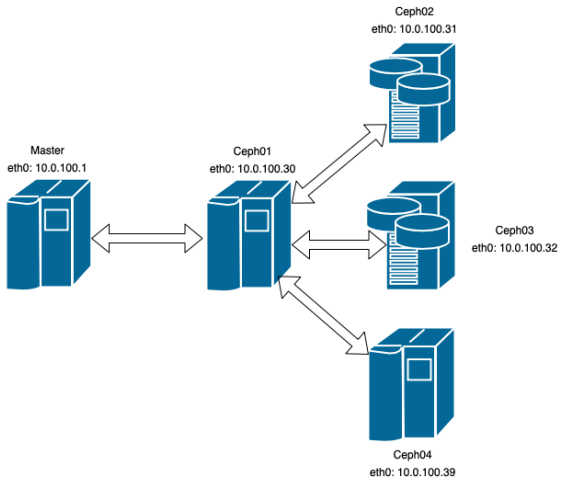

Figura 3.5-1

3.5.1 Instalación de Ceph

- En todos los Nodos

  - Generar las keygen en cada nodo

```
ssh-keygen
cat /root/.ssh/id_rsa.pub
```

  - Otorgar permisos de acceso en cada nodo, todos a todos

```
ssh-copy-id -i /root/.ssh/id_rsa.pub ceph01
ssh-copy-id -i /root/.ssh/id_rsa.pub ceph02
ssh-copy-id -i /root/.ssh/id_rsa.pub ceph03
ssh-copy-id -i /root/.ssh/id_rsa.pub ceph04
```

  - Editar el archivo /root/.ssh/config

```
Host ceph01
  Hostname ceph01
```

```
  User root

Host ceph02
  Hostname ceph02
  User root

Host ceph03
  Hostname ceph03
  User root

Host ceph04
  Hostname ceph04
  User root
```

- En el nodo servidor, en este caso estableceremos el nodo ceph01

  - Instalar ceph-deploy

```
wget -q -O-
'https://ceph.com/git/?p=ceph.git;a=blob_plain;f=keys/release.
asc' | apt-key add -

echo deb http://download.ceph.com/debian-jewel/ $(lsb_release
-sc) main | tee /etc/apt/sources.list.d/ceph.list

apt update
apt install ceph-deploy

mkdir cluster && cd cluster
```

  - Asignar el nodo monitor, en este ejemplo usaremos el nodo ceph01

```
ceph-deploy new ceph01
ceph-deploy mon create-initial
```

  - Lanzamos la instalación de ceph en todos los nodos

```
ceph-deploy install ceph01 \ ceph02 \ ceph03 \ ceph04
```

  - Añadir los discos al clúster de Ceph. Lo haremos desde el nodo ceph01

```
ceph-deploy disk zap ceph02:sdb
ceph-deploy disk zap ceph03:sdb

ceph-deploy --overwrite-conf osd prepare ceph02:sdb
```

```
ceph-deploy --overwrite-conf osd prepare ceph03:sdb

ceph-deploy osd activate ceph02:sdb1
ceph-deploy osd activate ceph03:sdb1
```

- Copiamos los archivos de configuración en todos los nodos y ponemos los permisos correctos a la clave privada.

```
ceph-deploy admin ceph01 ceph02 ceph03 ceph04

chmod +r /etc/ceph/ceph.client.admin.keyring

ssh ceph02
chmod +r /etc/ceph/ceph.client.admin.keyring

ssh ceph03
chmod +r /etc/ceph/ceph.client.admin.keyring

ssh ceph04
chmod +r /etc/ceph/ceph.client.admin.keyring
```

- Para usar Ceph Filesystem necesitamos un servidor de Metadatos, para ello lo instalaremos en el mismo servidor de monitorización ceph01.

```
ceph-deploy mds create ceph01

ceph mds stat
```

- Por último y en caso de reiniciarse un nodo OSD, añadimos los puntos de montaje en sus /etc/fstab para que al arrancar se vuelvan a conectar el clúster Ceph, en este caso los nodos 2 y 3

```
ssh ceph02
mount | grep osd | awk '{print $3}'`; echo "/dev/sdb1 $RES
xfs rw,noatime,attr2,inode64,noquota    0    0" >>
/etc/fstab

ssh ceph03
RES=`mount | grep osd | awk '{print $3}'`; echo "/dev/sdb1
$RES xfs rw,noatime,attr2,inode64,noquota    0    0" >>
/etc/fstab
```

- En el nodo cliente, en este caso el nodo ceph04

  - Creamos un nuevo dispositivo por bloques según el espacio que se tenga disponible, en este caso 5GB

    ```
    rbd create devceph --size 5000
    ```

  - Ejecutaremos la siguiente orden para evitar que se presente un error al mapear el disco y luego procederemos a mapearlo

    ```
    rbd feature disable devceph deep-flatten fast-diff object
    map exclusive-lock

    rbd map devceph
    ```

  - Con la siguiente orden podremos confirmar que el disco fue mapeado correctamente.

    ```
    rbd showmapped
    ```

  - Ahora ya podemos formatear y montar el dispositivo, como si fuera una partición de un disco duro en Unix

    ```
    mkfs.ext4 /dev/rbd/rbd/devceph
    mkdir -p /mnt/ceph
    mount -o discard /dev/rbd/rbd/devceph /mnt/ceph/
    ```

  - Para verificar que fue correctamente monta, ejecutamos la orden

    ```
    mount | grep ceph
    ```

3.5.2 Dificultades encontradas en Ceph

Durante la instalación de Ceph, siguiendo los pasos indicados de diferentes guías encontradas en la web, se presenta un error mostrando que no se han podido localizar los paquetes ceph-osd y ceph-mon como se muestra en la figura 3.5.2-1, debido a esto, el proceso de descarga e instalación se detiene, por lo que no fue posible finalizar la instalación para realizar la configuración y las pruebas con el sistema de almacenamiento Ceph.

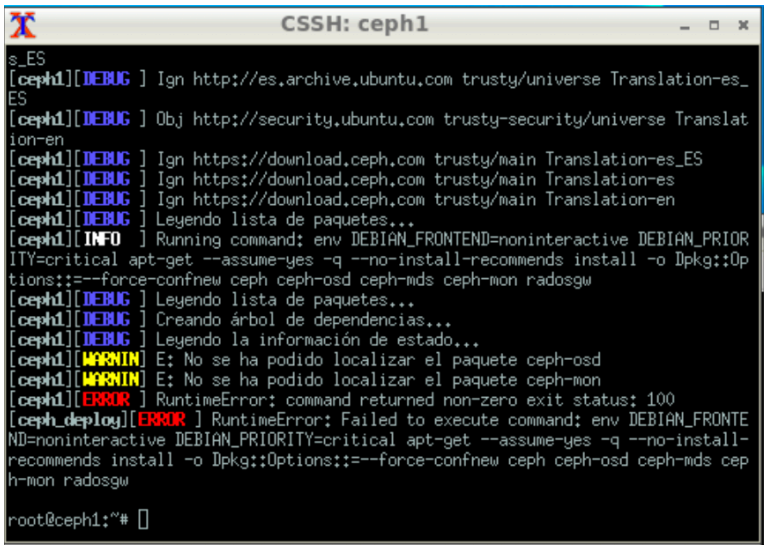

Figura 3.5.2-1

Se hicieron varios intentos con las últimas versiones Nautilus, Mimic, Luminous de Ceph y en todas las versiones se presentaba el mismo error. Finalmente se consiguió finalizar una instalación con la versión Jewel, como se muestra en las imágenes 3.5.2-2, 3.5.2-3, 3.5.2-4 y 3.5.2-5. Podemos ver como el proceso de instalación se ha completado en cada uno de los nodos sin presentar mensaje de error.

Figura 3.5.2-2

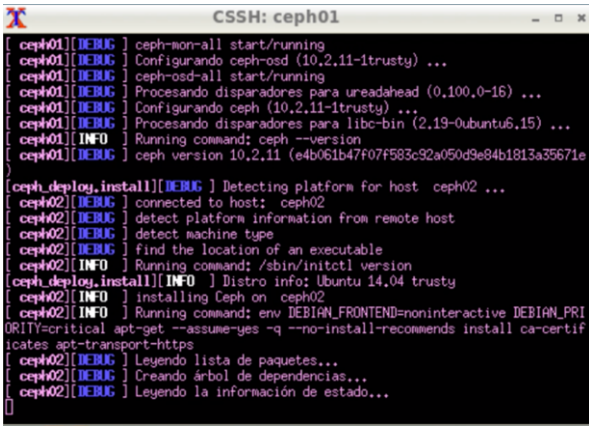

Figura 3.5.2-3

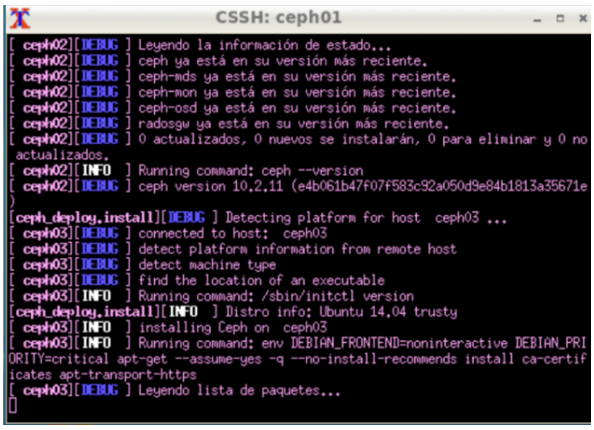

Figura 3.5.2-4

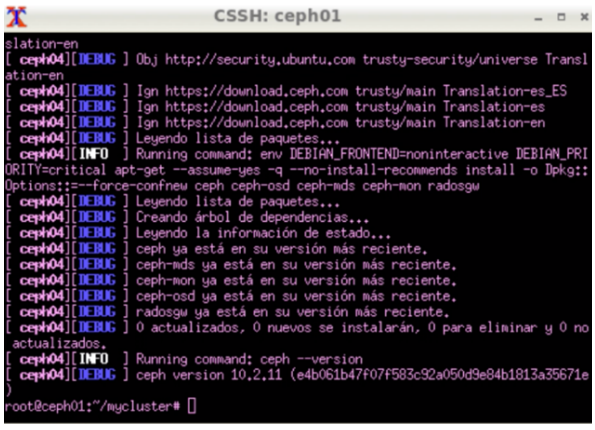

Figura 3.5.2-5

En los anteriores sistemas de almacenamiento se realizaron las instalaciones usando discos 5GB de capacidad. En el caso de Ceph, encontramos que al momento de hacer el montaje, el sistema Ceph crea dos particiones en el disco reservando 1GB para la configuración y el restante para el almacenamiento como podemos ver en la figura 3.5.2-6, razón por la cual, se aumento la capacidad de los discos a 6 GB para poder finalizar el montaje de discos durante la instalación de Ceph.

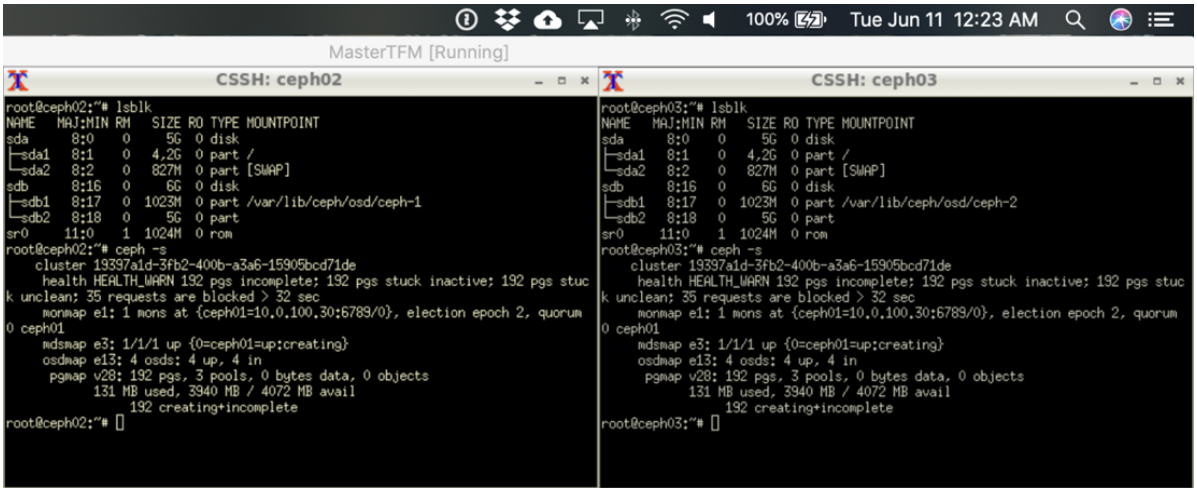

Figura 3.5-1

3.5.3 Pruebas en CEPH

Al realizar la prueba de lectura, lectura aleatoria, escritura, escritura aleatoria y, lectura y escritura simultaneas con la aplicación FIO y realizar una carga de archivos de 1 GB en CEPH se presentaron los siguientes resultados:

	1 thread		2 threads	
Read	65157 KB/s		72007 KB/s	
Rand read	51697 KB/s		65338 KB/s	
Write	28480 KB/s		33584 KB/s	
Rand write	40275 KB/s		49321 KB/s	
Read / write	62521 KB/s	23680 KB/s	68414 KB/s	29519 KB/s

3.6 Instalación y Pruebas de DRBD

Como se muestra en la figura 3.6-1 para la instalación y configuración del servicio DRBD hemos implementado 2 nodos, 1 servidor principal, 1 servidor secundario. Ambos servidores tendrán funciones de almacenamiento.

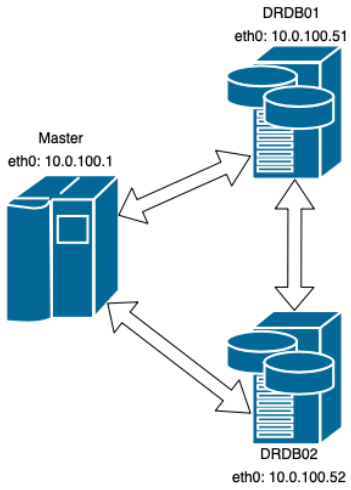

Figura 3.6-1

3.6.1 Instalación de DRBD

Para realizar la instalación y configuración de DRBD en los servidores se ejecutaron las siguientes órdenes.

En ambos servidores:

- Editar el archivo /etc/hosts

```
10.0.100.51 DRBD01
10.0.100.52 DRBD02
```

- Se actualiza Unix y se instalan las aplicaciones necesarias

```
apt update
apt install -y drbd8-utils drbdlinks
apt install -y ntp
```

- Editar el archivo /etc/drbd.d/global_common.conf

```
global {
      usage-count yes;
}
…

startup {
degr-wfc-timeout 60;
}
…

net {
cram-hmac-alg sha1;
shared-secret “TFM”;
protocol C;
}
```

- Creamos un archivo en la carpeta /etc/drbd.d/ en donde se indicarán las configuraciones de los nodos a usar, en este caso, el archivo se llamara r0.res

```
resource r0 {
```

```
      on DRBD01 {
      device /dev/ drbd0;
      disk /dev/sdb1;
      address 10.0.100.51:7788;
      meta-disk internal;
      }
      on DRBD02 {
      device /dev/drbd0;
      disk /dev/sdb1;
      address 10.0.100.52:7788;
      meta-disk internal;
      }
}
```

- Se ejecutan las siguientes órdenes

```
modprobe drbd
drbdadm create-md r0
drbdadm up r0
```

- Se ejecutan las siguientes órdenes (solo en el primer nodo)

```
drbdadm -- --overwrite-data-of-peer primary r0
drbdadm connect r0
drbdadm adjust r0
```

- Para comprobar que la instalación fue correcta se ejecuta la orden

```
cat /proc/drbd
drbd-overview
```

- Se realiza el montaje de disco (solo en el primer nodo)

```
mkfs.ext4 /dev/drbd0
mkdir -p /mnt/drbd
mount /dev/drbd0 /mnt/drbd
```

3.6.2 Dificultades encontradas en DRBD

Al realizar estos pasos de instalación se pudo comprobar que los nodos quedaron en comunicación, se ejecutaron las órdenes para que el nodo primario pase a ser el

secundario y el secundario el primario y los nodos se mantenían en correcto funcionamiento.

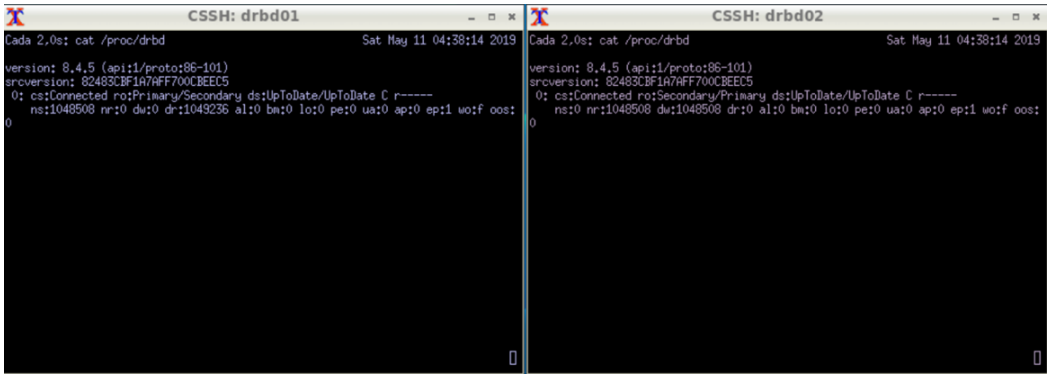

Figura 3.6.2-1

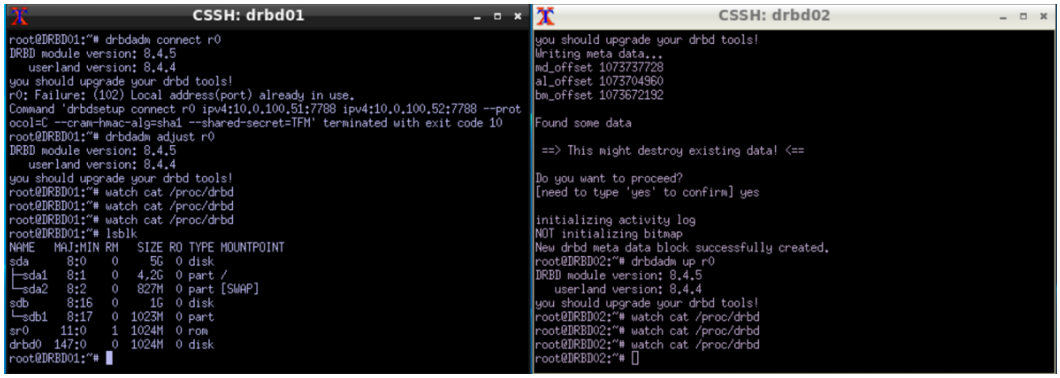

Figura 3.6.2-2

Sin embargo, al reiniciar los nodos, ambos servidores buscan al otro servidor durante el arranque como se muestra en las imágenes 3.6.2-3 y 3.6.2-4, por lo que se le indica a los nodos que salten el proceso para continuar con el arranque.

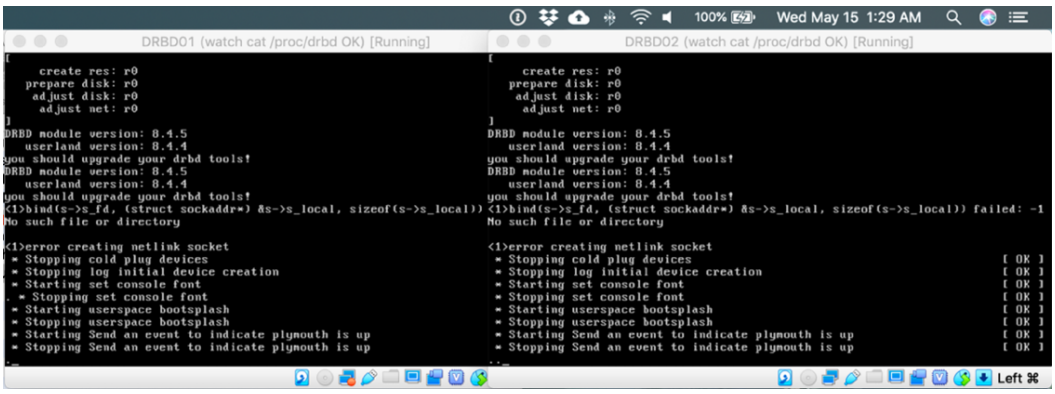

Figura 3.6.2-3

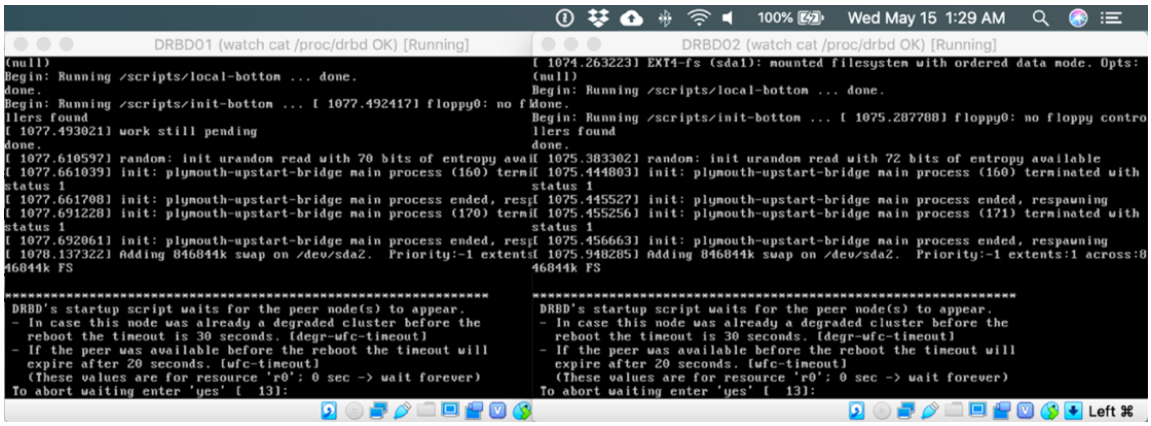

Figura 3.6.2-4

Por esta razón ambos servidores al terminar de iniciar quedan como nodos secundarios, esto se pudo confirmar ejecutando la orden “watch cat /proc/drbd” en ambos nodos.

También encontramos que durante la instalación, configuración y ejecución de órdenes durante las pruebas en DRBD, el sistema muestra un mensaje indicando la posibilidad de actualizar DRBD de la versión 8.4.4 a 8.4.5.

Se intentó realizar esta actualización mediante la orden “apt update” de Ubuntu, pero al realizar la actualización, el sistema indica que no existen actualizaciones pendientes, también se intento de manera manual y el sistema de almacenamiento perdió la estabilidad presentando errores en su funcionamiento, por esta razón se mantuvo la versión 8.4.4.

3.6.3 Pruebas en DRBD

Al realizar la prueba de lectura, lectura aleatoria, escritura, escritura aleatoria y, lectura y escritura simultaneas con la aplicación FIO y realizar una carga de archivos de 1 GB en RAID 5 se presentaron los siguientes resultados:

	1 thread		2 threads	
Read	17655 KB/s		19213 KB/s	
Rand read	11787 KB/s		16725 KB/s	
Write	25168 KB/s		38729 KB/s	
Rand write	26314 KB/s		36500 KB/s	
Read / write	31575 KB/s	13624 KB/s	36485 KB/s	15738 KB/s

También se hicieron algunas pruebas estableciendo el nodo DRBD01 como primario y luego desconectar el nodo DRBD02 que se encontraba como secundario. Se pudo comprobar que al desconectar el nodo DRBD02, el primario se mantiene funcionando, pudiendo tener acceso a los archivos.

Al reiniciar el nodo DRBD02 se ejecutó la orden cat /proc/drbd y encontramos que el nodo DRBD02 estaba realizando algunas tareas, al finalizar las tareas en proceso, se procedió a invertir los papeles de los nodos estableciendo el nodo DRBD02 como primario y al acceder a los datos encontramos que los cambios realizados en el nodo DRBD01 mientras el nodo DRBD02 se encontraba apagado ya estaban disponibles.

3.7 Evaluación

Como se mencionó anteriormente las pruebas de velocidad de lectura y escritura se realizaron con el software FIO. Como podemos observar en las figuras 3.7-1, 3.7-2, 3.7-3, 3.7-4 y 3.7-5 los sistemas de almacenamiento en GlusterFS, MooseFS y Ceph presentan mejores prestaciones de lectura y escritura en la mayoría de las pruebas con respecto a los otros sistemas evaluados. Esto se debe al paralelismo que ofrecen, bien permitiendo que los archivos se almacenen en varios dispositivos, ofreciendo acceso concurrente a varios clientes.

En los resultados de RAID 1 y RAID 5 se puede ver que los rangos de diferencia en cada prueba de uno y dos hilos tienden a mantenerse en cada una de las evaluaciones.

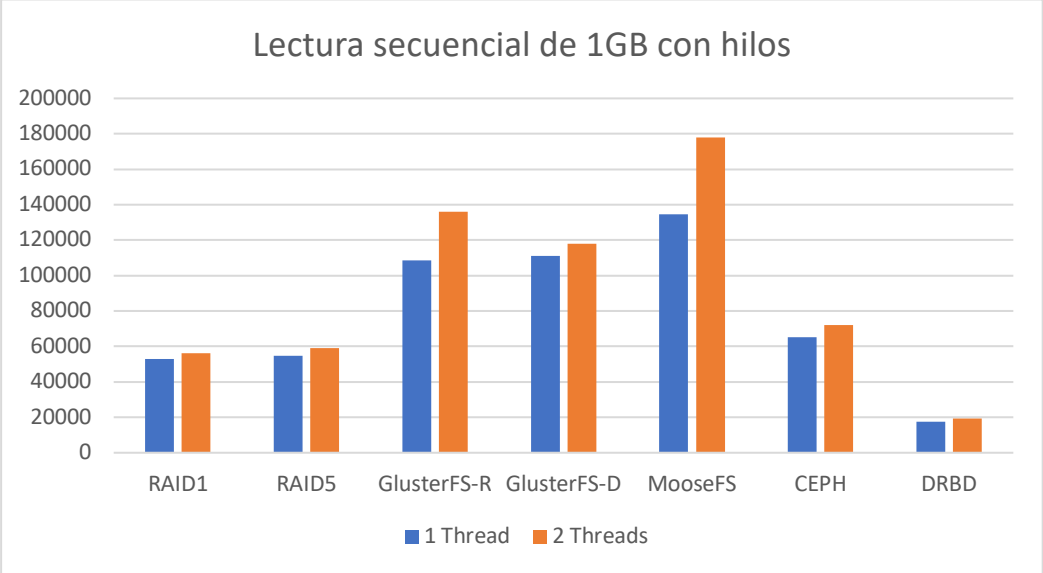

Figura 3.7-1

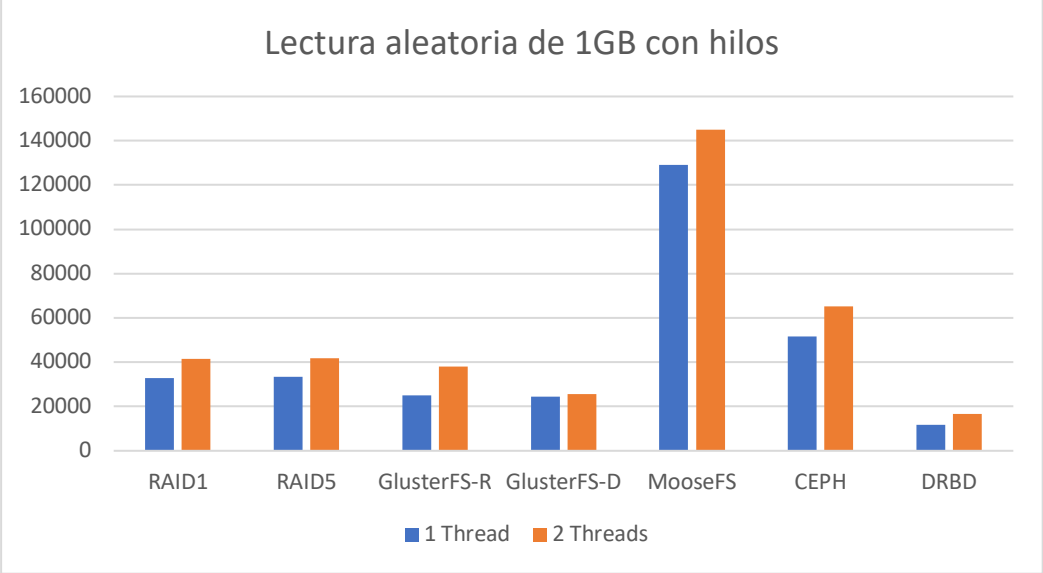

Figura 3.7-2

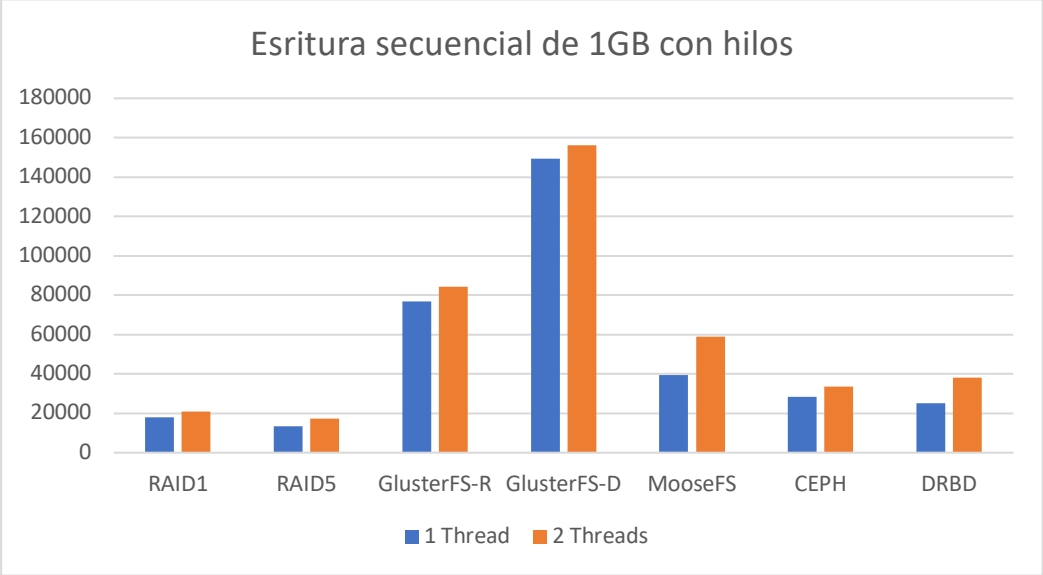

Figura 3.7-3

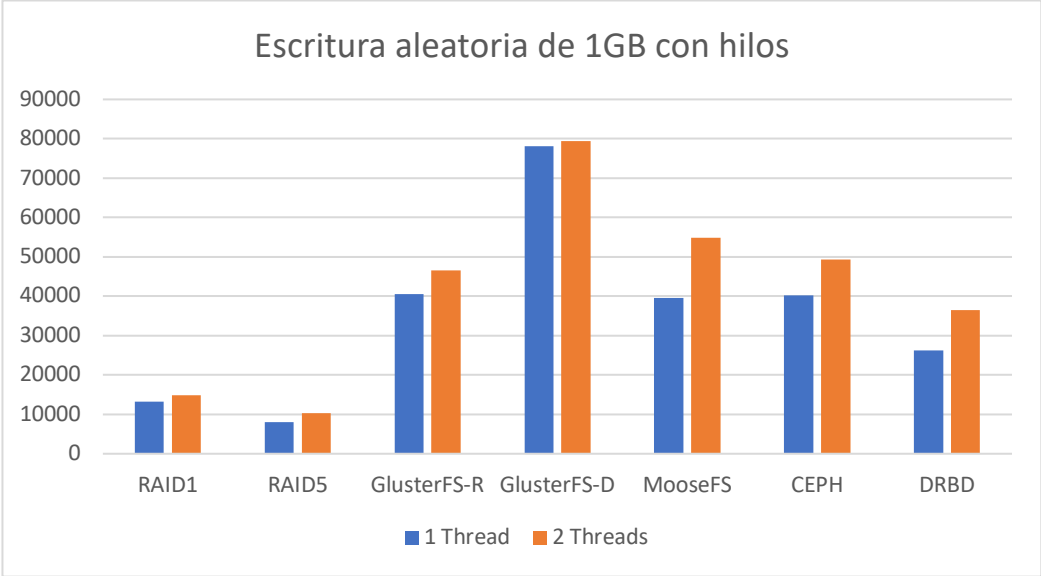

Figura 3.7-4

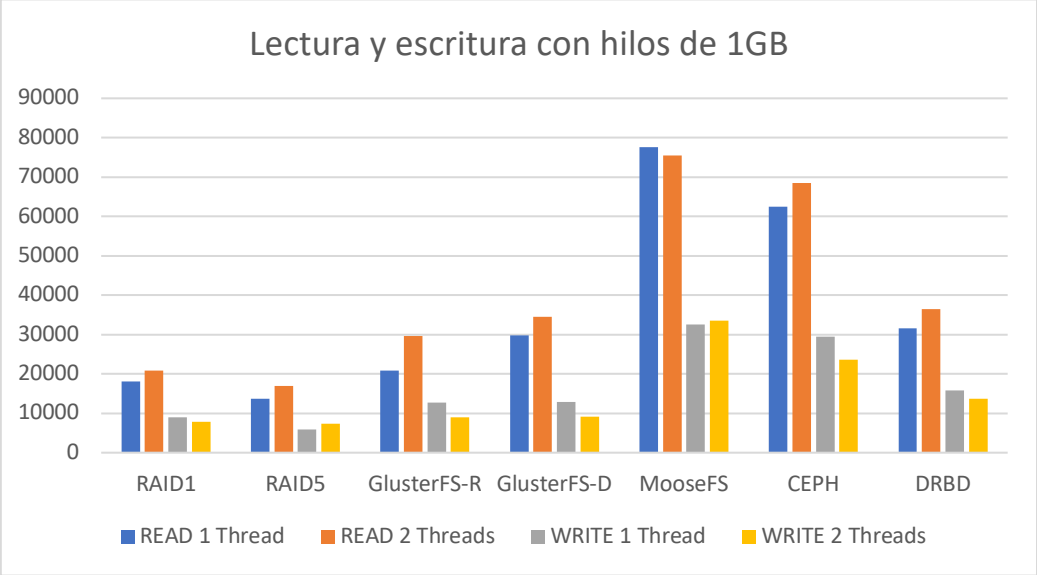

Figura 3.7-5

También debemos tomar en cuenta que estos resultados podrían no ser del todo realistas ya que actualmente los tiempos de lectura y escritura dependen también del número de clientes con acceso a los sistemas de almacenamiento, las velocidades de lectura y escritura que ofrezcan los dispositivos a nivel de hardware, además de que las pruebas fueron realizadas en maquinas virtuales con hardware limitado.

Por lo que los resultados podrían ser muy diferentes si se realizaran estas mismas pruebas en diferentes ambientes y condiciones.

4. CONCLUSIONES

Para observar como funcionan cada uno de los sistemas de almacenamiento, se realizaron sencillas pruebas de almacenamiento de archivos y se pudo notar que son eficientes confiables y estables, también pudimos notar que se pueden instalar de manera sencilla si se siguen los pasos correctamente.

Si no se tiene experiencia con ninguno de ellos, puede llevar un poco de trabajo conseguir que funcionen correctamente, pero una vez instalados no son difíciles de administrar.

Se podría decir que GlusterFS y MooseFS son los más sencillos de configurar y utilizar además de ser estables. MooseFS seria el más manejable, ya que cuenta con una interfaz gráfica a través de una aplicación CGI que facilita la administración permitiendo ver los estados y detalles de las configuraciones en cada unos de los nodos de forma general.

Si no se requiere un sistema de almacenamiento robusto o existen restricciones de hardware, la opción ideal seria trabajar con varios discos en NAS o DRBD por sus bajos requerimientos de hardware y su facilidad de instalación y uso.

5. BIBLIOGRAFIA

[1] «Redes Zone,» [En línea]. Available: https://www.redeszone.net/2019/03/12/tipos-raid-nas/.

[2] «clouding.io,» [En línea]. Available: https://clouding.io/kb/compartir-directorios-en-gnulinux-con-nfs/.

[3] D. R. Gralin, «Universidad Simon Bolivar,» 1998. [En línea]. Available: https://ldc.usb.ve/~emilio/Portafolio/Exposiciones/nfs.pdf.

[4] M. Viera, «mviera.io,» [En línea]. Available: https://mviera.io/blog/compartiendo-ficheros-con-glusterfs/ .

[5] «proxadmin.es,» [En línea]. Available: https://www.proxadmin.es/blog/glusterfs-almacenamiento-distribuido/.

[6] «Gluster,» [En línea]. Available: https://docs.gluster.org/en/latest/.

[7] «eslared.org.ve,» [En línea]. Available: http://www.eslared.org.ve/walc2012/material/track5/GlusterFS.pdf.

[8] «MooseFS,» [En línea]. Available: https://moosefs.com.

[9] «Manual MooseFS,» [En línea]. Available: https://moosefs.com/Content/Downloads/moosefs-3-0-users-manual.pdf.

[10] «MooseFS Support,» [En línea]. Available: https://moosefs.com/support/.

[11] «MooseFS,» [En línea]. Available: https://moosefs.com/about/#advantages.

[12] «jhosman.com,» [En línea]. Available: https://jhosman.com/index.php/2016/04/29/instalar-y-configurar-un-3nodo-con-ceph-y-ubuntu/.

[13] «Ceph,» [En línea]. Available: http://docs.ceph.com/docs/master/start/intro/.

[14] «ceph.com,» [En línea]. Available: https://ceph.com/geen-categorie/incremental-snapshots-with-rbd/.

[15] «docs.ceph.com,» [En línea]. Available: http://docs.ceph.com/docs/jewel/rbd/rbd-snapshot/.

[16] «Universidad La Laguna,» [En línea]. Available: https://www.ull.es/servicios/stic/2016/11/11/sistema-de-almacenamiento-ceph/.

[17] «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Distributed_Replicated_Block_Device.

[18] «redesteleco.com,» [En línea]. Available: https://redesteleco.com/drbd-raid1_en_red_entre_varios_equipos/.

[19] «es.slideshare.net,» [En línea]. Available: https://es.slideshare.net/adfc7/drbd.

[20] «Unixmen,» [En línea]. Available: https://www.unixmen.com/linux-basics-create-network-bonding-on-ubuntu-14-10/.

[21] «Academia,» [En línea]. Available: https://www.academia.edu/30894682/Clusterización_Proxmox.