Nube de almacenamiento distribuida con GlusterFS: Volumen distribuido y replicado

*Sistemas Concurrentes y Distribuídos

Coronado Huamán Ayrton Fabio
Escuela de Ciencia de la Computación
Facultad de Ciencias-UNI
Lima, Perú
acoronadoh@uni.pe

Flores Huamaní Luis Angel
Escuela de Ciencia de la Computación
Facultad de Ciencias-UNI
Lima, Perú
lfloresh@uni.pe

Barrientos Porras Herlees Brayan
Escuela de Ciencia de la Computación
Facultad de Ciencias-UNI
Lima, Perú
herlees.barrientos.p@uni.pe

***Abstract*—Este documento es el estudio y implementación de una nube de almacenamiento distribuído en Ubuntu Linux 20.04 del tipo volúmen distribuído replicado con GlusterFS.**

***Index Terms*—nube, almacenamiento, distribuído, volúmen, replicado, GlusterFS, Ubuntu.**

I. Introducción

Vivimos en un mundo donde la información esta creciendo de forma impredecible y almacenarla en un lugar centralizado pueda generar muchos problemas, no solo problemas de gestión de información, también problema de seguridad, logística, baja eficiencia y productividad.
Ante esta problemática el almacenamiento distribuído surge como una solución ideal.
En este artículo tenemos como objetivo estudiar, diseñar e implementar este tipo de sistemas mediante el software gratuito y Open Source GlusterFS, con un servidor desarrollado en Nodejs.

A. Objetivos

- Implementar una nube de almacenamiento distribuida para guardar archivos de los usuarios.

B. Objetivos específicos

- Configurar una red de nodos en un servidor de almancemiento virtualizado.
- Implementar un cluster de almacenamiento con Glusterfs.
- Desarrollar un sistema de comunicación entre los clientes y el servidor principal.
- Desarrollar una aplicación web para el cliente.

II. Estado del arte

A. Implementación de una Arquitectura Tecnológica basada en Cloud Computing como soporte al portafolio de proyectos profesionales de la EISC [4]

Tiene como objetivo implementar una arquitectura tecnológica utilizando la tecnología de prestación de servicios de Cloud Computing que soporte los proyectos profesionales de la Escuela de Ingeniería de Sistemas y Computación (EISC). El proyecto contiene investigación de los conceptos relacionados a Cloud Computing, contemplando definiciones, historia, modelos, tipos de soluciones y metodologías utilizadas en implementaciones de arquitectura.
Según los autores el proyecto obtuvo el visto bueno de consultores externos, los costos fueran principalmente de recursos huamanos ,ya que el software empleado era gratuito y la infraestructura fue proveida por una empresa privada.

B. Características de la Nube

- Tercerización de recursos. - El proveedor de la nube asume la responsabilidad para la adquisición de hardware y mantenimiento de este.
- Utilidad informática. - El cliente solicita recursos adicionales, según sea necesario, y de manera similar libera estos recursos cuando no se necesitan de acuerdo a sus demandas de autoservicio.
- La escalabilidad y la autogestión. - Las nubes son como programadores libres que se ocupan de los problemas de escalabilidad, estas nubes ofrecen el cambio de tamaño de una manera automática de los recursos de hardware virtualizados. La escalabilidad y la autogestión es más simple a través de un único dominio administrativo, pero puede surgir ciertos problemas.
- Alta disponibilidad .- De acuerdo a los SLA que se establezcan con el proveedor, se pueden asegurar hasta un 99.95% de disponibilidad.
- Recuperación de desastres .- Esta recuperación ante desastres en la nube son copias de seguridad y restauración estratégicas que implica el almacenamiento y mantenimiento de copias de los registros que se dan en un entorno Cloud como medida de seguridad.

C. Modelos de Despliegue

- Cloud pública. - Plataforma abierta al público en general. Puede ser administrada mediante una empresa, alguna entidad académica o gubernamental, o alguna mezcla de

ellas. Las instalaciones se encuentran en las instalaciones del proveedor.

- Cloud privada. - La infraestructura de la nube privada está estructurada para el uso exclusivo de una organización que puede ser utilizada por más de una unidad de negocio. No obstante, la gestión y operación puede estar encargada de la misma organización o algún tercero que se encuentre tanto en los interiores o exteriores de las instalaciones
- Cloud híbrida. - La infraestructura está conformada por más de un tipo de Cloud. Aprovecha el beneficio que ambos tipos de Cloud brindan con la finalidad de manejar la carga de trabajo de la organización.
- Cloud comunitaria. - Modelo que consiste en dos o más tipos de Cloud. Utilizada para cubrir necesidades en común de una comunidad de organizaciones en específico. Puede ser gestionada u operada por uno o más empresas tanto en los interiores o exteriores.

III. Herramientas

A. GluterFS

GlusterFS es un sistema de archivos de almacenamiento escalable y distribuido conectado a la red que le permite agrupar recursos de almacenamiento de varias máquinas. A su vez, esto permite manipular diversos dispositivos de almacenamiento que se distribuyen entre muchas computadoras como una unidad única y más potente.

B. NodeJS

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

C. Ubuntu 20.04

Ubuntu es un sistema operativo de software libre y código abierto. Es una distribución de Linux basada en Debian. Actualmente corre en computadores de escritorio y servidores. Está orientado al usuario promedio, con un fuerte enfoque en la facilidad de uso y en mejorar la experiencia del usuario. Actualmente la versión 20.04 estable, es la que contiene los paquetes mas actualizados de GlusterFS.

D. Flutter

Flutter es un SDK de código fuente abierto de desarrollo de aplicaciones móviles creado por Google. Suele usarse para desarrollar interfaces de usuario para aplicaciones en Android, iOS y Web.

IV. Modelo de investigación

A. Conceptos de Glusterfs

[2] En Glusterfs se tienen tres conceptos fundamentales:

- 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.
- Volumen distribuido [1]: Los archivos se distribuyen en varios bricks en el volumen, de forma que el archivo 1 sólo podrá almacenarse en un único brick, por lo que no habrá redundancia de datos. Este tipo de volumen distribuido hace que sea más fácil y barato escalar el tamaño del volumen. No obstante, 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.
- Volumen replicado [1]: 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 otros bricks. El número de réplicas se configura por el usuario al crear el volumen, si queremos dos réplicas necesitaremos al menos dos bricks, si queremos tres réplicas necesitaremos tres bricks, y así sucesivamente. 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.

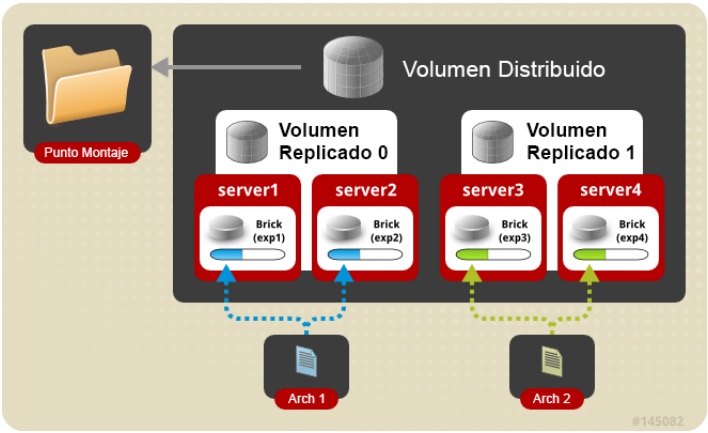

Fig. 1. Volumen distribuido y replicado

B. Concurrencia en Nodejs

Node.js funciona con un modelo de evaluación de un único hilo de ejecución, usando entradas y salidas asíncronas las cuales pueden ejecutarse concurrentemente en un número de hasta cientos de miles sin incurrir en costos asociados al cambio de contexto. Este diseño de compartir un único hilo de ejecución entre todas las solicitudes atiende a necesidades de aplicaciones altamente concurrentes, en el que toda operación que realice entradas y salidas debe tener una función callback.

C. Modelo a desarrollar

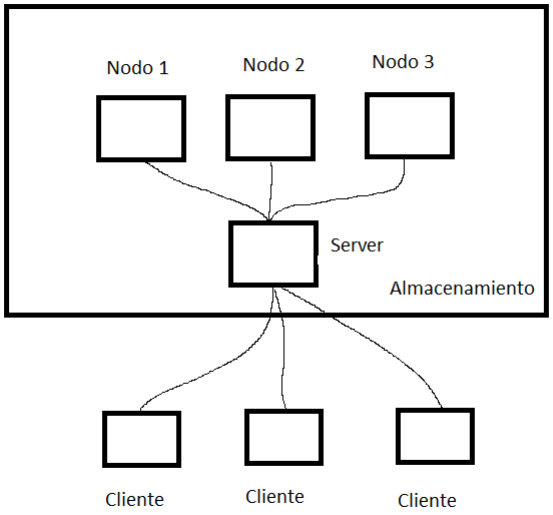

Fig. 2. Implementación

Cada nodo contiene un brick, el servidor tendrá el montaje de almacenamiento.

V. Implementación [3]

EL primer paso para implementar este sistema es configurar un cluster, para ello vamos a usar virtualbox:

A. Cluster

1) Virtual Box: Para comenzar debemos descargar la imagen ISO de ubuntu Server 20.04 y crear 4 máquinas virtuales, para este caso hemos creado una y la hemos clonado. El usuario de la máquina creada es gfs1, luego establecemos el adaptador como puenteado para que la máquina virtual sea considerada como una mas de nuestra red. Debemos asignarles ips estáticas desde nuestro router para que la configuración no se pierda. Para este caso he asignado las siguientes direcciones ip:

- Nodo 1 : 192.168.1.120
- Nodo 2 : 192.168.1.121
- Nodo 3 : 192.168.1.123
- Cliente: 192.168.1.122

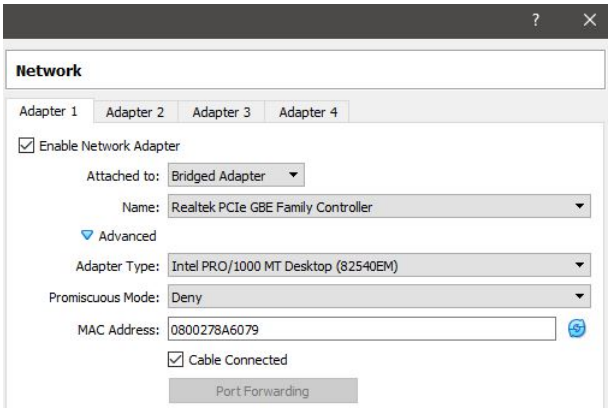

Fig. 3. Red en Virtual box

2) Ubuntu: Luego debemos configurar el hostname para cada máquina editando el archivo: /etc/hostname. Para este caso tenemos tres máquinas que serán parte del cluster y otra donde será montada la unidad. La configuración es la siguiente:

- Nodo 1 : gfs1
- Nodo 2 : gfs2
- Nodo 3 : gfs3
- Cliente: client

Luego para facilitar la configuración debemos agregar todos estos nombres a cada máquina, editando el archivo /etc/hosts. Ejemplo para el Nodo1:

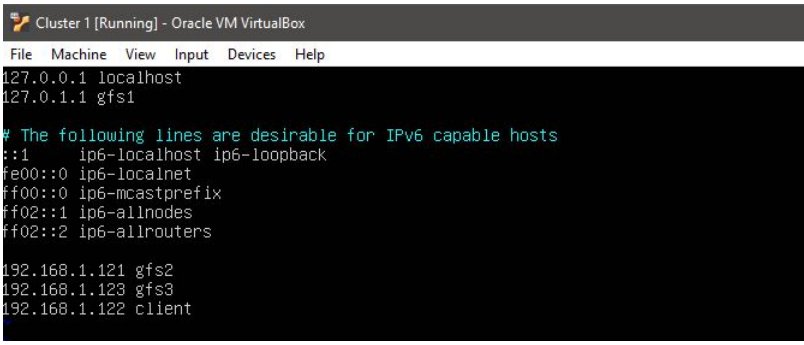

Fig. 4. Archivo /etc/hosts para gfs1

Debemos instalar en cada Nodo GlusterFs para ello seguimos la siguiente rutina:

- sudo apt update
- sudo apt install software-properties-common
- sudo add-apt-repository ppa:gluster/glusterfs-7
- sudo apt install glusterfs-server

Finalmente comprobamos que el servicio este ejecutandose con:

- sudo systemctl status glusterd.service

3) Firewall: Ahora debemos configurar el firewall de ubuntu para poder establecer comunicaciónes entre ellos y crear un grupo de almacenamiento. Se sabe que el servicio de Gluster usa el puerto 24007 por lo que debemos permitir el acceso a ese nodo de almacenamiento.

Para permitir ello debemos ejecutar los siguientes comandos el gfs1:

- sudo ufw allow from gfs2_ip_address to any port 24007
- sudo ufw allow from gfs3_ip_address to any port 24007
- sudo ufw allow from client_ip_address to any port 24007

Finalmente para que ninguna otra máquina pueda acceder por este puerto.

- sudo ufw deny 24007

De manera similar en gfs2 y gfs3.

4) Glusterfs: Para establecer la comunicación entre los nodos, debemos ejecutar el comando en alguno de los nodos. En este caso escogi gfs1:

- sudo gluster peer probe gfs2
- sudo gluster peer probe gfs3

Estos comandos le dice el nodo gfs1 que confie en gfs2 y gfs3. Si todo va bien el mensaje de salida debe ser:

- peer probe: success

Podemos probar con el comando:

- sudo gluster peer status

La salida debe ser algo como esta:

```
gfs1@gfs1:~$ sudo gluster peer status
[sudo] password for gfs1:
Number of Peers: 2

Hostname: gfs2
Uuid: 82a5b452-2b2a-4638-a81d-df78e7affa76
State: Peer in Cluster (Connected)

Hostname: gfs3
Uuid: e563d534-e7c6-4855-9c08-7c5affd12422
State: Peer in Cluster (Connected)
```

Fig. 5. Salida del comando sudo gluster peer status

Ahora debemos crear los volúmenes de almacenamiento. Primero creamos una carpeta en la raíz llamada gluster-storage en cada uno de los nodos, luego debemos ingresar el comando desde cualquiera de los nodos. En mi caso escogi gfs1: Volumen de replicación

- sudo gluster volume create volume1 replica 3 gfs1:/gluster-storage gfs2:/gluster-storage gfs3:/gluster-storage force

Luego iniciamos el volumen

- sudo gluster volume start volume1

Finalmente podemos ver el estado del volumen con:

- sudo gluster volume status

```
[sudo] password for gfs1:
Status of volume: test-volume
Gluster process                                 TCP Port  RDMA Port  Online  Pid
------------------------------------------------------------------------------
Brick gfs1:/mnt/sdb1/bricks/test-volume         49152     0          Y       1088
Brick gfs2:/mnt/sdb1/bricks/test-volume         49152     0          Y       837
Brick gfs3:/mnt/sdb1/bricks/test-volume         49152     0          Y       829
Self-heal Daemon on localhost                   N/A       N/A        Y       1110
Self-heal Daemon on gfs2                        N/A       N/A        Y       886
Self-heal Daemon on gfs3                        N/A       N/A        Y       877

Task Status of Volume test-volume
------------------------------------------------------------------------------
There are no active volume tasks
```

Fig. 6. Salida del comando sudo gluster volume status

Nota: En este caso la carpeta creada esta en un disco duro diferente.

Ahora necesitamos permitir las conexiones desde el cliente en los nodos, para ello ejecutamos el siguiente comando en cada un de los nodos:

- sudo ufw allow from client_ip_address to any port 49152
- sudo ufw deny 49152

5) Cliente: Ahora vamos a configurar el cliente Para ello debemos ejecutar el siguiente comando:

- sudo apt install glusterfs-client

Debemos crear un volumen de montaje en el cliente. En este caso vamos a crear la carpeta en la raíz llamada replicate y montarla con:

- sudo mount -t glusterfs gfs1:volume_name /replicate/

Podemos verificar el montaje con el comando df:

```
gfs1@client:~$ df
Filesystem                         1K-blocks     Used Available Use% Mounted on
udev                                  458756        0    458756   0% /dev
tmpfs                                 100488     1056     99432   2% /run
/dev/mapper/ubuntu--vg-ubuntu--lv    9219412  5307252   3424124  61% /
tmpfs                                 502424        0    502424   0% /dev/shm
tmpfs                                   5120        0      5120   0% /run/lock
tmpfs                                 502424        0    502424   0% /sys/fs/cgroup
/dev/sda2                             999320   201080    729428  22% /boot
/dev/loop0                             56320    56320         0 100% /snap/core18/1880
/dev/loop2                             73088    73088         0 100% /snap/lxd/16099
/dev/loop4                             30720    30720         0 100% /snap/snapd/8790
/dev/loop1                             56704    56704         0 100% /snap/core18/1885
/dev/loop5                             30720    30720         0 100% /snap/snapd/8542
/dev/loop3                             72320    72320         0 100% /snap/lxd/16922
gfs1:/vol-1                        27658236 15180480  11290224  58% /distributed
gfs1:/test-volume                  18438824 10343672   7303464  59% /replicate
tmpfs                                 100484        0    100484   0% /run/user/1000
```

Fig. 7. Salida del comando df

6) Api: Ahora desarrollamos una Api que nos permita guardar los archivos en el volumen montado. Para ello usamos node.js con distintos paquetes:

- adm-zip: Lo usamos para comprimir un directorio
- express: Es un marco de aplicación web para Node.js, lanzado como software gratuito y de código abierto bajo la Licencia MIT
- cors: Proporciona un middleware Connect/Express que se puede usar para habilitar CORS con varias opciones.
- express-fileupload: Middleware express simple para cargar archivos.
- fs: sistema de archivos simplificado que nos permite crear carpetas y guardar archivos en el cluster.

Hemos implementado distintas funcionalidades para interactuar con el cluster.

Funcionalidades implementadas:

- Crear carpeta.
- Subir archivo.
- Descargar archivo o carpeta.
- Obtener contenido de la carpeta.

Finalmente debemos desplegar el api en el cliente del cluster

7) Interfaz usuario: Para desarrollar la interfaz hemos usado el sdk de google Flutter que va a usar las funcionalidades de la api que hemos creado. Para ello usamos métodos y un Widget llamado FutureBuilder para solicitar la información del contenido de las carpetas cuando la petición al cliente del cluster este lista.

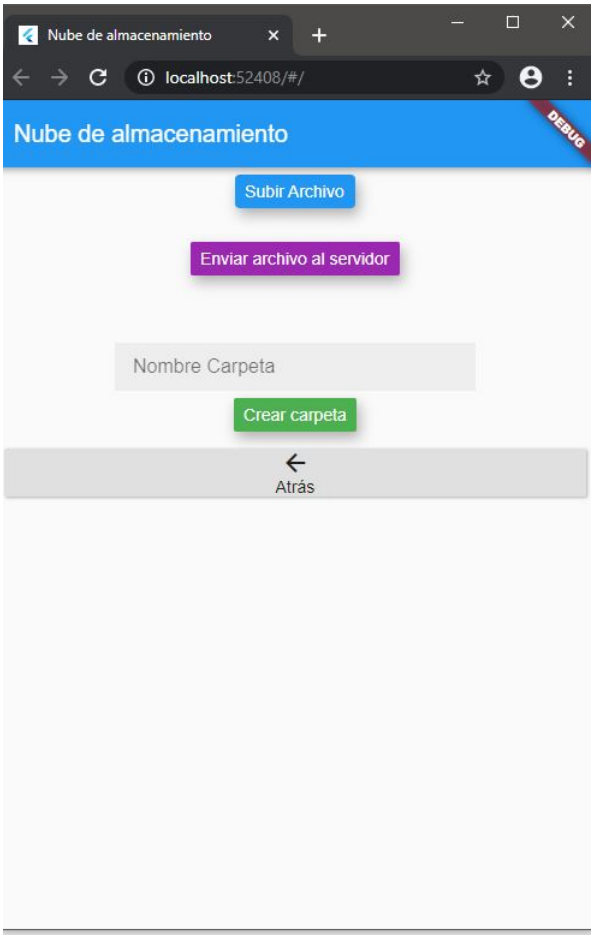

Fig. 8. Interfaz de usuario

Prueba en un maquina con 4 despligue de clientes subiendo archivos.

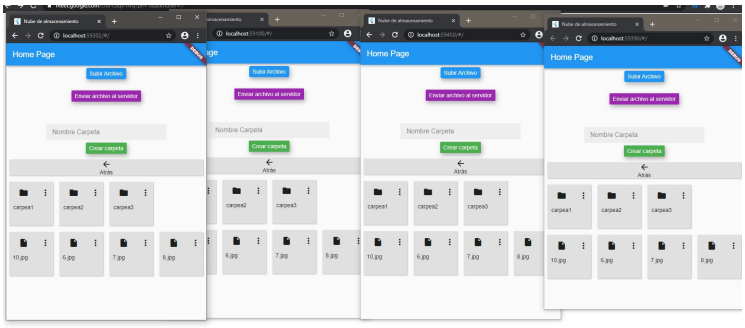

Fig. 9. Prueba de 4 clientes enviando archivos al mismo tiempo

VI. Conclusiones

- Crear un volumen de replicación y distribución con gluster es altamente escalable ya que soporta una gran cantidad de nodos.
- La implementación realizada facilmente se puede replicar a un entorno real. Donde cada nodo podría estar en diferentes partes del mundo.
- Implementar un volumen distribuido-replicado necesita alrededor de 6 nodos para ver su funcionamiento.

References

[1] CHAPTER 6. RED HAT GLUSTER STORAGE VOLUMES https://access.redhat.com/documentation/en-us/red_hat_gluster_storage/3.1/html/administration_guide/chap-red_hat_storage_volumes
[2] GLUSTERFS: CREA TU ALMACENAMIENTO DISTRIBUIDO https://www.proxadmin.es/blog/glusterfs-almacenamiento-distribuido/
[3] HOW TO CREATE A REDUNDANT STORAGE POOL USING GLUSTERFS ON UBUNTU 18.04 https://www.digitalocean.com/community/tutorials/how-to-create-a-redundant-storage-pool-using-glusterfs-on-ubuntu-18-04
[4] Implementación de una Arquitectura Tecnológica basada en Cloud Computing como soporte al portafolio de proyectos profesionales de la EISC https://repositorioacademico.upc.edu.pe/bitstream/handle/10757/623029/Liz%C3%A1rraga_lr.pdf?sequence=5&isAllowed=y