UNIVERSIDAD NACIONAL DE INGENIERIA

Recinto Universitario Simón Bolívar

Facultad de Electrotecnia y Computación

Trabajo monográfico para optar al Título de Ingeniero en Computación

"DISEÑO DE SISTEMA DE RESPALDO DE DATOS EN LÍNEA MEDIANTE CLÚSTER EN ARREGLO REDUNDANTE DE DISCOS VIRTUALES (RAVD) PARA RED KANGAROO, S.A."

Autores:

Br. Roberto José Blandino Cisneros

Br. Nathanael Ismael Alemán Ramírez

Tutor:

Ing. José Leónidas Díaz Chow

Managua, octubre de 2022

DEDICATORIA

Esta Tesis está dedicada:

Principalmente a Dios, por ser nuestro creador, Padre y Señor, por darnos la fuerza y valentía para continuar con este proceso de obtener uno de los anhelos más deseados de todo estudiante de carrera universitaria.

A nuestros padres: José Roberto Blandino Obando y Natalia Mercedes Cisneros Salvatierra; y, en memoria de Daysi Mary Ramírez Stuart y Abraham Morazán Monterrey, quienes con su amor, esfuerzo y paciencia nos ayudaron a salir adelante en nuestros estudios.

A nuestras esposas Katherin Prissila Sevilla Zelaya y Mayara Arguello Astorga quienes siempre han estado a nuestro lado animándonos, y alentándonos en los momentos más difíciles, sin su ayuda incondicional este trabajo no hubiese sido posible.

A la memoria de doña Gloria Ramírez Stuart, abuelita de Nathanael, quien vivió su vida, actuando de una manera concienzudamente aferrada a sus creencias, ayudando tanto a sus familiares, como a personas conocidas o desconocidas, las cuales necesitaran ayuda tanto en lo material, como en lo espiritual. Ella hasta el último momento que Dios le permitió estar con vida, dio fe del gran amor que sentía por su Señor Jesucristo, porque estaba segura de que ni aún la muerte iba a poder apartarla de su lado. Muchas gracias por enseñarnos que Dios está siempre con nosotros, peleando nuestras batallas como poderoso gigante.

Finalmente queremos dedicarle este trabajo a nuestro tutor, Ing. José Díaz Chow, por ser la persona que ha estado ayudándonos en todo este tiempo de investigación, y redacción final del documento, él sin duda alguna ha sido un pilar fundamental en la realización de este trabajo monográfico.

AGRADECIMIENTOS

Agradecemos principalmente a Dios por bendecirnos en cada día de nuestras vidas, por guiarnos a lo largo de nuestra existencia, y por ser el apoyo y fortaleza incondicional en aquellos momentos de dificultad y de debilidad.

A nuestros padres: José Roberto Blandino Obando y Natalia Mercedes Cisneros Salvatierra; y, en memoria de Daysi Mary Ramírez Stuart y Abraham Morazán Monterrey, por ser los principales promotores de nuestros sueños, por creer y confiar en nuestro potencial, por los consejos, valores, y principios que nos han inculcado. Gracias porque durante los años de universidad, cada uno de ellos nos facilitaron todas las comodidades de su hogar, cuando uno de nosotros necesitaba quedarse estudiando o haciendo uso de la computadora para la realización de los trabajos asignados, muchas gracias por el ejemplo de esfuerzo y valentía, y enseñarnos a no temer a las adversidades y tiempos de sequía.

A nuestras esposas Katherin Prissila Sevilla Zelaya y Mayara Arguello Astorga, quienes siempre tuvieron fe en nosotros, nos han ayudado en todo momento, sea de felicidad, tristeza o momentos de duelo; gracias por alentarnos y animarnos con todo su amor para culminar este trabajo monográfico, somos un equipo juntos, gracias, porque sin su ayuda no hubiéramos llegado a la meta.

Agradecemos a nuestra Alma Mater UNI por ofrecernos las herramientas y laboratorios para nuestro aprendizaje, y crecimiento intelectual. Agradecemos especialmente a nuestros docentes de la FEC (Faculta de Electrotecnia y Computación) de la UNI (Universidad Nacional de Ingeniería), por habernos compartido sus conocimientos a lo largo de todos los años de preparación de nuestra carrera, gracias porque con su esfuerzo ejemplo y paciencia han logrado no solo formar profesionales, sino también gente de bien, con valores morales y éticos, los cuales son igual de importantes que el conocimiento técnico que todo profesionista pueda adquirir a lo largo de su vida.

Finalmente queremos agradecer de manera muy especial a nuestro tutor Ing. José Díaz Chow, quien, con su dirección, conocimiento, enseñanza, y correcciones permitió el desarrollo y culminación de este trabajo, muchas gracias por toda la ayuda brindada en todo este tiempo.

RESUMEN

Red Kangaroo S.A. (REKASA) es una empresa que presta servicios de Web Hosting y virtualización, los cuales requieren el alojamiento y respaldo de datos de los clientes. En vista de que en este momento para ellos no es factible adquirir soluciones de almacenamiento propietarias por ser muy costosas, REKASA requiere implementar un sistema de respaldo de datos de bajo costo que sea estable, altamente disponible, escalable y en línea, el cual le asegure los datos de sus clientes ante posibles fallos de la infraestructura tecnológica.

A fin de suplir las necesidades de REKASA, en este trabajo monográfico se diseñó un sistema de respaldo de datos en línea basado en un clúster de almacenamiento en arreglo redundante de discos virtuales, denominado por los autores RAVD (siglas de Redudant Array of Virtual Disks), sobre legacy hardware, el cual es una solución de ingeniería que integra diferentes tecnologías open Source trabajando de manera orquestada para proveer la funcionalidad necesaria con los atributos requeridos por la empresa.

Para certificar que el modelo satisface las necesidades de REKASA, se implementó un prototipo funcional, cuyo desempeño fue evaluado mediante una serie de pruebas de control: a) pérdida de un elemento del arreglo en frio y en caliente, b) pérdida del controlador maestro, donde está montado el arreglo de discos virtuales, c) incremento de memoria interna para medir el rendimiento en el desempeño del clúster y d) incremento del volumen en línea.

A partir de los resultados de las pruebas realizadas se demostró que el modelo RAVD diseñado satisface los requerimientos establecidos y da solución al problema de almacenamiento de datos de REKASA de manera satisfactoria.

INDICE DE CONTENIDO

Págs.

ÍNDICE DE FIGURAS

Págs.

INDICE DE CUADROS

Págs.

REFERENCIA DE SIGLAS, ACRÓNIMOS Y ABREVIATURAS

SIGLA	SIGNIFICADO o CONCEPTO
API	Application Programming Interface: en español «Interfaz de Programación de Aplicaciones»
APT	Advance Packaging Tool: en español <<Herramienta de Empaquetado Avanzado>>
BSD	Berkeley Software Distribution o BSD: en español «distribución de software Berkeley»
CD	Compact Disc: en español «Disco Compacto»
CIFS	Common Internet File System: en español «Sistema de Archivo Común de Internet»
CPN	Controller Physical Node: en español <<Nodo físico controlador>>
CPU	Central Processing Unit: en español «Unidad Central de Procesamiento»
CVN	Controller Virtual Node: en español <<Nodo virtual controlador>>
DAS	Direct Attached Storage: en español «Almacenamiento de Conexión Directa»
DRBD	Distributed Replicated Block Device: en español «Dispositivos de Bloque de Replicación de Distribución»
DVD	Digital Versatile Disc or Digital Video Disc: en español «Disco Versátil Digital»
E/S	Entrada/Salida o E/S: en inglés «Input/Output or I/O»
EB	Exabytes: medida estándar en bytes cuyo valor es 1024^6
ESXI	Elastic Sky X Integrated: nombre de producto del hipervisor desarrollado por VMWare
FLOPS	Floating-point Operations Per Second: en español «Operaciones de Coma Flotante por Segundo»
FTP	File Transfer Protocol: en español «Protocolo de Transferencia de Ficheros»
GB	Gigabyte: medida estándar en bytes cuyo valor es 1024^3
GNU	Acrónimo recursivo de "GNU's Not Unix" / "Gnu Not Unix": en español «GNU no es Unix»
GPU	Graphical Processing Unit: en español «Unidad de Procesamiento Gráfico»
HA	High Availability: en español «Alta Disponibilidad»
HAL	Hardware Abstraction Layer: en español «Capa de Abstracción de Hardware»
HD	Hard Drive: en español <<Disco duro>>
HP	Hewlett-Packard: nombre de marca de fabricante
I/O	Input and Output: en español <<Entrada y Salida>>
IBM	International Business Machines Corp.: en español «Maquina de Negocios Internacionales»
IEC	International Electrotechnical Commission: en español <<Comisión Internacional Electrotécnica>>
IP	Internet Protocol: en español «Protocolo de Internet»
ISO	International Organization for Standardization: en español «Organización Internacional de Normalización»
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission: en español «Organización Internacional de Normalización/Comisión Electrotécnica Internacional»
KDE	K Desktop Environment: en español «Entorno de Escritorio K»
LACP	Link Aggregation Control Protocol: en español «Protocolo de control de enlaces agregados»

SIGLA	SIGNIFICADO o CONCEPTO
LAHF	Load AH Flags: en español <<Carga AH de Bandera>> (AH es el nombre del REGISTRO en el CPU)
LAN	Local Area Network: en español «Red de Área Local»
LVM	Logical Volume Manager: en español «Administrador de Volúmenes Lógicos»
MAN	Manual: aplicación en sistemas UNIX para desplegar ayuda y documentación de comandos
MB	Megabyte: medida estándar en bytes cuyo valor es 1024^2
MD4	Message-Digest Algorithm 4: en español «Algoritmo de Resumen del Mensaje 4»
MD5	Message-Digest Algorithm 5: en español «Algoritmo de Resumen del Mensaje 5»
MDADM	Multiple Device Administrator: en español «Administrador de Múltiples Dispositivos»
MIT	Massachusetts Institute of Technology: en español <<Instituto Tecnológico de Massachusetts>>
NAS	Network Attached Storage: en español «Administrador de Dispositivos Múltiples»
NBD	Network Block Device: en español «Dispositivo de Bloque de Red»
NBDC	Network Block Device Client: en español «Controlador de Dispositivo de Bloque de Red»
NBDS	Network Block Device Server: en español «Servidor de Dispositivo de Bloque de Red»
NFS	Network File System: en español «Sistema de Archivos de Red»
NIC	Network Interface Card: en español «Tarjeta de Interfaz de Red»
NIST	National Institute of Standards and Technology: en español <<Instituto Nacional de Estándares y Tecnología>>
OS	Operative System: en español «Sistema Operativo»
OSI	Open System Interconnection: en español «Modelo de Interconexión de Sistemas Abiertos»
PID	Process ID: en español «Identificador de Procesos»
RAID	Redundant Array of Independent Disks: en español «Arreglo Redundante de Discos Independientes»
RAM	Random Access Memory: en español «Memoria de Acceso Aleatorio»
RAVD	Redundant Array of Virtual Disks: en español «Arreglo Redundante de Discos Virtuales»
REKASA	Red Kangaroo Sociedad Anónima
RFC	Request for Comments: en español <<Solicitud de comentarios>>
S.A.	Sociedad Anónima
SAHF	Store AH into Flags: en español: <<Almacén AH en Bandera>> (AH es el nombre del REGISTRO en el CPU)
SAN	Storage Area Network: en español «Red de Área de Almacenamiento»
SASL	Simple Authentication and Security Layer: en español «Capa de Seguridad y Autenticación Simple»
SHA1	Secure Hash Algorithm: en español «Algoritmo de Hash Seguro»
SNIA	Storage Networking Industry Association: en español <<Asociación Industrial de Almacenamiento en Red>>
SPN	Storage Physical Node: en español <<Nodo Físico de Almacenamiento>>
STP	Spanning Tree Protocol: en español <<Protocolo de Árbol de Expandido>>
SVN	Storage Virtual Node: en español <<Nodo Virtual de Almacenamiento>>

SIGLA	SIGNIFICADO o CONCEPTO
TCP	Transmission Control Protocol: en español <<Protocolo de Control de Transmisión>>
TCP/IP	Transmission Control Protocol/Internet Protocol: en español «Protocolo de Control de Transmisión/Protocolo de Internet»
TFTP	Trivial File Transfer Protocol: en español «Protocolo de Transferencia de Archivos Trivial»
TLS	Transport Layer Security: en español «Seguridad de la Capa de Transporte»
UDP	User Datagram Protocol: en español «Protocolo de Datagrama de Usuario»
UML	Unified Modeling Language: en español «Lenguaje de Modelado Unificado»
UNIX	Uniplexed Information and Computing System: en español <<Sistema de computación e información uniplexado>>
UOC	Universitat Oberta de Catalunya: en español «Universidad Abierta de Cataluña»
URL	Uniform Resource Locator: en español «Localizador Uniforme de Recursos»
USB	Universal Serial Bus: en español «Bus Universal en Serie»
VIM	Visual editor Improved: en español <<Editor Visual Mejorado>>
VM	Virtual Machine: en español «Máquina Virtual»
VMM	Virtual Machine Monitor: en español «Monitor de Máquina Virtual»
VPN	Virtual Private Network: en español «Red Privada Virtual»
VSFTP	Very Secure FTP Daemon: en español <<Demonio Muy Seguro FTP>>
XFS	High-performance 64-bit journaling file system: en español «Sistema de Archivos de 64 bits con Journaling de Alto Rendimiento»

CAPITULO I. GENERALIDADES

I.1. Introducción

La informática y los productos de información que el ser humano genera a partir de ella avanzan a una velocidad vertiginosa. Los centros de cómputo van creciendo más y más, así como también los datos que éstos albergan. Hilbert y López (Hilbert & López, 2011) calcularon que al 2007, el 94% de la información que había generado la humanidad a ese entonces, unos 295 EB[1] (exabytes) en total, estaba en formato digital. A inicios de 2011, en solo tres años, esa cantidad de información prácticamente se había duplicado, calculándose en unos 600 EB, de los cuales el 99.9% es en formato digital. Dada la volatilidad de los datos digitales y la gran cantidad de ellos en existencia, se evidencia la primordial importancia y necesidad de respaldarlos apropiadamente.

Red Kangaroo S.A. (REKASA) es una empresa de tecnología de la información y las comunicaciones con fuerte presencia en el país, que ofrece una amplia cartera de servicios. REKASA requiere de un sistema de respaldo de datos de bajo costo que sea estable, escalable y en línea, para asegurar los datos de sus clientes ante cualquier falla eventual en la infraestructura tecnológica que soporta sus servicios.

El presente trabajo monográfico comprende el diseño de un prototipo funcional de un sistema de respaldo de datos en línea mediante el uso de un clúster[2] de

---

[1] **EB:** Notación para Exabyte, unidad grande de almacenamiento de datos informáticos, equivalente en el sistema internacional de medidas a 10^{18} bytes. El prefijo exa-, adoptado en 1991, procede del griego ἕξα, que significa «seis» que corresponde a la cantidad de múltiplos de miles que incorpora la cantidad. Su equivalente en el sistema binario (EiB) es 2^{60} bytes.

[2] **Clúster:** es un grupo de ordenadores vinculados entre sí para trabajar colaborativamente. (Ver sección II.5)

almacenamiento basado en un Arreglo Redundante de Discos Virtuales (RAVD[3]), así como la evaluación de su desempeño, para determinar si es apropiado para satisfacer las necesidades de esta empresa.

I.2. Antecedentes

Red Kangaroo, S. A. (REKASA) desde sus inicios, ha tenido por lema dedicarse a desarrollar soluciones adaptables a las necesidades de sus clientes. Ofrece servicios de Web Hosting[4] y virtualización[5], diseño e implementación de redes de datos de alto desempeño, y planificación e implementación de Datacenters[6]. Los servicios de Web Hosting y virtualización requieren el alojamiento y respaldo de datos de los clientes que se sirven en aplicaciones en línea, por tanto, varían y crecen en volumen a cada momento, en tiempo real.

Para suplir las necesidades de respaldo de estos datos, REKASA ha usado en el pasado, con resultados aceptables, medios de respaldo tradicionales como discos externos y servidores Network File System (NFS[7]). Las principales limitantes de estos medios son: el espacio reducido, dificultad de manejo y la falta de escalabilidad. Los discos externos no pueden estar siempre en línea y requirieren

---

[3] **RAVD:** siglas de "Redundant Array of Virtual Disks", **concepto introducido por los autores en este trabajo monográfico**, a partir del bien establecido concepto de RAID (Redundant Array of Independent Disks).

[4] **Web Hosting:** Servicio de alojamiento web que provee a los usuarios de Internet un sistema para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía web.

[5] **Virtualización:** es una tecnología basada en software que permite ejecutar múltiples sistemas operativos y aplicaciones simultáneamente en un mismo servidor. (Ver detalles en sección II.11)

[6] **Datacenters:** lugar perfectamente acondicionado para albergar cientos de servidores, con temperatura y humedad constantes, con altísimos niveles de seguridad y restricción de entrada, y conectado a Internet por medio de múltiples operadores internacionales de telecomunicaciones.

[7] **NFS (Network File System):** el Sistema de Archivo de Red es un protocolo de nivel de aplicación, según el Modelo OSI, que posibilita que distintos sistemas conectados a una misma red accedan a ficheros remotos como si se tratara de locales. Ver detalles en referencia (Nowicki, 1989)

considerable tiempo y esfuerzo para localizar y restaurar datos. El respaldo en NFS puede ser en línea, pero se limita al espacio del disco en el servidor. Estas restricciones obligan a que cada cierto tiempo se deba adquirir nuevos discos externos y gestionarse la administración de lo que se almacena en el servidor NFS mediante la eliminación de respaldos con fechas antiguas, para dar espacio al almacenamiento de los nuevos respaldos.

Asimismo, al establecer los términos contractuales de servicio, se indica que REKASA es responsable de la integridad, consistencia y confidencialidad de los datos de los clientes, al menos por ahora, no puede depender de terceros para el respaldo y aseguramiento de éstos, por lo que, para enfrentar el requerimiento de respaldar el creciente volumen de datos, la empresa debe adoptar una solución de almacenamiento de respaldo propia y ser manejados en una nube de almacenamiento interno para mantener seguro el contenido de los mismos.

I.3. Justificación

REKASA requiere implementar un sistema de respaldo de datos de bajo costo que sea estable, escalable y en línea, para asegurar los datos de sus clientes ante la eventualidad de alguna falla en la infraestructura hardware que soporta los servicios.

Las soluciones de almacenamiento de red comerciales, como la red de almacenamiento de área local (SAN)[8], o los almacenamientos adjuntos de red (NAS)[9] son sumamente costosas para el alcance actual de servicios de REKASA,

[8] **Red de almacenamiento de área local (SAN):** consiste en una red de área local de dispositivos de almacenamiento. Ver detalles en referencia (SAN, 2015)

[9] **Almacenamientos adjuntos de red (NAS):** consiste en sistemas de almacenamiento que se pueden conectar directamente a la red local de computadoras para ser accedidos por éstas. Ver detalles en referencia (NAS, 2015).

y el lento retorno económico de estas implementaciones no justifica la inversión en estos momentos.

REKASA requiere una solución de respaldo que emplee software libre o gratuito y que permita usar en ella parte de la infraestructura de servidores existentes, que no tienen tanto poder de cómputo para servir aplicaciones al público, pero que tienen bastante espacio de almacenamiento, por lo que podrían servir excelentemente para respaldo.

El diseño y (su prototipo) desarrollado en este trabajo de monografía, consiste en un clúster de almacenamiento en alta disponibilidad sobre un arreglo redundante de discos virtuales (RAVD)[10], el cual satisface las necesidades de REKASA, integrando los atributos que se requieren, sin los altos costos de las soluciones propietarias comerciales, proveyendo los siguientes beneficios:

1. **Estabilidad,** gracias a la disponibilidad que proporciona la clusterización[11] y la redundancia de discos.
2. **Escalabilidad**, al ser implementado en software sobre discos virtuales, permitiendo el crecimiento del almacenamiento en línea.
3. **Gestión y manejo del arreglo** desde línea de comandos, en línea, en caliente y en tiempo real[12].
4. **Menor tiempo de respuesta** que las soluciones empleadas anteriormente (NFS y discos externos).

---

[10] **RAVD:** Siglas de Redundant Array of Virtual Disks, término acuñado por los autores de este trabajo monográfico a partir de la similitud de la solución propuesta con los RAID (Redundant Array of Independent Disks) que está basada en discos reales.

[11] **Clusterización:** un clúster es el conjunto de dos o más computadoras (llamados nodos o miembros) que trabajan juntos para realizar una tarea. Ver detalles en sección sección II.6.

[12] Con ciertas limitaciones en caso de la operación de escritura/lectura, la cual puede requerir reinicio de la operación en ciertos escenarios.

5. **Permite emplear hardware heterogéneo** en su implementación, debido a que se construye sobre máquinas virtuales, permitiendo a REKASA utilizar sus servidores legacy actualmente ociosos en la implementación de la solución de respaldo.
6. **Drástica reducción de costos** en comparación con los requeridos si se implementara una solución propietaria, debido a que emplea solamente software libre o de uso gratuito sobre el hardware ocioso de la empresa.

I.4. Objetivos

I.4.1. Objetivo General

Diseñar un sistema de respaldo de datos en línea basado en un clúster de almacenamiento en arreglo redundante de discos virtuales (RAVD) que satisfaga los requerimientos de REKASA.

I.4.2. Objetivos Específicos

a) Seleccionar las técnicas y tecnologías de software libre o gratuito disponibles para implementar un clúster en arreglo redundante de discos virtuales que pueda emplearse como base para el diseño del sistema de respaldo.
b) Diseñar la propuesta de sistema de respaldo empleando las técnicas y tecnología de software seleccionadas.
c) Crear un prototipo funcional del sistema de respaldo en línea basado en un clúster de almacenamiento en arreglo redundante de discos virtuales (RAVD) que permita validar el diseño.
d) Evaluar el desempeño del prototipo a fin de verificar si el sistema propuesto satisface los requerimientos de REKASA y es apropiado como vía de solución a sus necesidades de respaldo de datos.

I.5. Metodología para la organización del trabajo

El presente trabajo monográfico consiste en generar una solución de respaldo como **producto de integración de tecnologías** *open source* a fin de que trabajen de manera orquestada, para suplir adecuadamente los requerimientos de respaldo de datos en línea de bajo costo para REKASA, empleando un clúster de almacenamiento basado en un arreglo de discos virtuales sobre legacy hardware[13]. A dicho modelo los autores denominan con las siglas RAVD (Redudant Array of Virtual Disks)[14].

A fin de comprobar la validez y nivel de desempeño de la solución propuesta, se realizó un prototipo funcional, al cual se aplicaron pruebas de evaluación con el propósito de determinar la efectividad de la solución, su rendimiento, fiabilidad y disponibilidad, ante la manipulación de cuatro eventos:

a. **Pérdida de un miembro del subclúster de almacenamiento en frio y en caliente.**
b. **Pérdida del controlador maestro del subclúster de control donde está montado el arreglo de discos virtuales.**

---

[13] **Legacy hardware**: se refiere a equipo que por su longevidad ya no es apropiado para servir aplicaciones de última generación. Depende del momento en que el hardware fue creado o lanzado al mercado. Un hardware es considerado legacy cuando su fecha de salida a producción es muy distante con respecto al sistema operativo de última generación que se quiera instalar, perdiendo así muchas funcionalidades que el sistema operativo ofrece por el hecho de que el hardware ha dejado de tener soporte por su propio fabricante en las nuevas tecnologías de software liberadas recientemente.

[14] **RAVD** es un término acuñado por los autores en referencia a la similitud del modelo propuesto con RAID (Redundant Array of Independent Disk). Este término no fue encontrado en la revisión documental realizada en Internet, como tampoco el uso de este conjunto de tecnologías en la configuración empleada en el presente trabajo, como propuesta de solución de almacenamiento de datos, por lo cual se considera que es un aporte novedoso al cuerpo de conocimiento del campo de la tecnología de la información.

c. **Incremento de memoria interna para medir el rendimiento en el desempeño del clúster.**
d. **Incremento del volumen en línea.**

El desarrollo del trabajo se estructuró en cinco etapas, las cuales se llevaron a cabo de la siguiente manera:

Figura (01) - Etapas del proceso metodológico para la organización del trabajo monográfico

Etapa 1. Revisión de información: Se obtuvo la información disponible de métodos y técnicas de virtualización, clusterización y almacenamiento, mediante un proceso de recopilación y revisión de literatura relacionada con tópicos de trabajos referentes y los recursos disponibles en internet.

Etapa 2. Selección de métodos, técnicas y tecnologías: Sobre la base de la información recopilada, se seleccionaron los métodos, técnicas y tecnologías a utilizar en el diseño de la solución.

Etapa 3. Diseño: Se elaboró el diseño del modelo de clúster de almacenamiento de arreglo de discos virtuales a ser construido.

Etapa 4. Implementación: Se llevó a cabo la materialización de la idea a un modelo funcional, mediante la instalación y configuración de las aplicaciones y scripts necesarios para proveer la funcionalidad del clúster de almacenamiento virtual redundante.

Etapa 5. Evaluación y perfeccionamiento: Se realizaron las pruebas de funcionalidad de la solución, se implementaron las mejoras requeridas y se analizó en qué medida el sistema propuesto satisface los requerimientos de REKASA.

CAPITULO II. MARCO CONCEPTUAL

II.1. Tecnologías de Almacenamiento

Diferentes autores (Miller & Fadden, 2014), (Tate, Bernasconi, Mescher, & Scholten, 2003) coinciden en clasificar las tecnologías de almacenamiento empresarial de datos en tres tipos:

II.1.1. Direct Attached Storage (DAS)

Es el método de almacenamiento más común y sencillo. Consiste en conectar el dispositivo de almacenamiento directamente al servidor o estación de trabajo, es decir, físicamente conectado al dispositivo que hace uso de él (DAS, 2015).

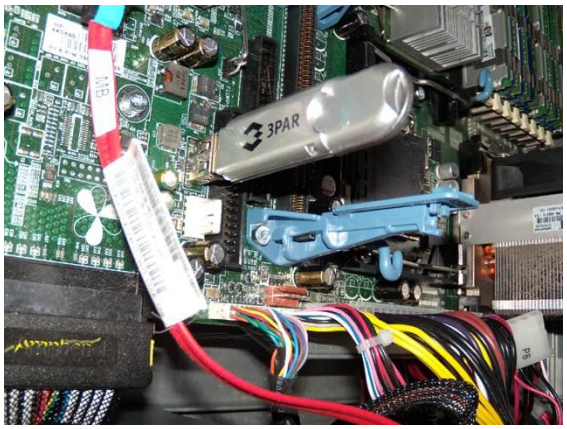


Figura (02) - DAS

II.1.2. Storage Area Network (SAN)

Según lo definen (Poelker & Nikitin, 2014) y (Chidlow, 2003), es un método que implementa su propia red de dispositivos de almacenamiento que generalmente no son accesibles a través de la red de área local (LAN) por otros dispositivos.

Una SAN consiste en una infraestructura de comunicaciones, que ofrece conexiones físicas, y una capa de administración, que organiza las conexiones, dispositivos de almacenamiento y los sistemas informáticos de manera que la

transferencia de datos es segura y robusta **Fuente especificada no válida.**. Entre sus propiedades más comunes están la disponibilidad, la consolidación y la seguridad. Los datos son lo más valioso de toda empresa, y la SAN es una herramienta actualmente indispensable para toda empresa que genera datos electrónicos y que necesite el respaldo y almacenamiento de estos. Es muy usual que una SAN esté acompañada del protocolo NFS (Network File System) para la distribución de los datos a través de la red.

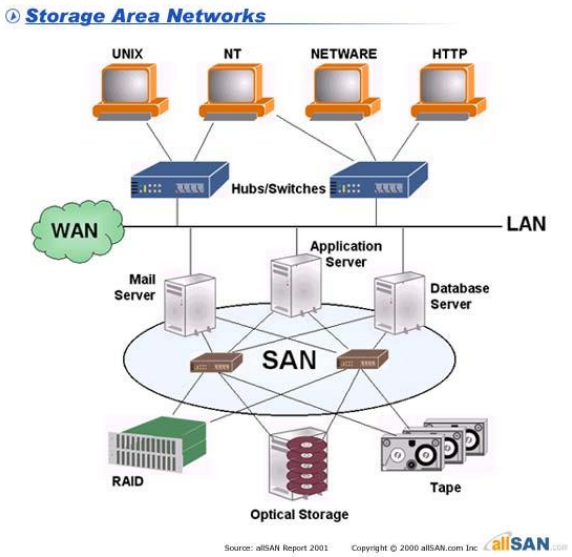


Figura (03) - SAN

II.1.3. Network Attached Storage (NAS)

Es el nombre dado a una tecnología de almacenamiento dedicada a compartir la capacidad de almacenamiento de un computador (Servidor) que ofrece servicios a computadoras personales o servidores clientes a través de una red (normalmente TCP/IP[15]), haciendo uso de un Sistema Operativo optimizado para dar acceso con

15 **TCP/IP:** el modelo TCP/IP describe un conjunto de guías generales de diseño e implementación de protocolos de red específicos para permitir que un equipo pueda comunicarse en una red de extremo a extremo especificando como los datos deben ser formateados,

los protocolos CIFS[16], NFS, FTP[17] (File Transfer Protocol) o TFTP[18] (Trivial File Transfer Protocol) (NAS, 2015).

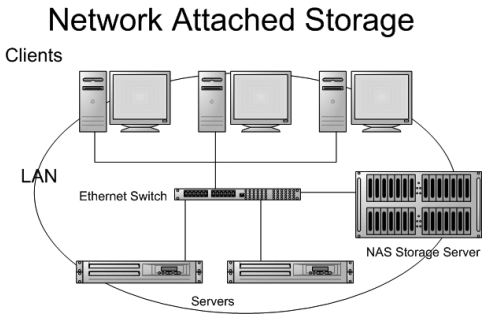


Figura (04) - NAS

II.2. Tecnologías para replicar y publicar dispositivos de bloques en red.

II.2.1. Distributed Replicated Block Device (DRBD)

Esta tecnología es un software que permite hacer una réplica remota entre dos dispositivos de bloques físicos ubicados en redes distintas. Su funcionamiento es similar al modelo RAID 1 (ref. II.5. Redundancia en el almacenamiento: RAID), a diferencia de que, en lugar de realizar un espejo entre dos discos locales, se realiza un espejo entre dos discos ubicados cada uno en lugares remotos.

direccionados, transmitidos, enrutados y recibidos por el destinatario. Ver detalles en referencia (Socolofsky & Kale, 1991)

16 CIFS: es una forma estándar que usan computadoras bajo sistema operativo Windows para compartir archivos en la intranet e internet. Ver detalles en referencia (CIFS, 2015)

17 FTP: Protocolo para la transferencia de archivos entre distintos ordenadores. Ver detalles en referencia (Postel & Reynolds, 1985)

18 TFTP: es un protocolo de transferencia de archivos parecido al FTP, pero en una versión muy limitada, orientado a realizar transferencias de archivos pequeños. Ver detalles en referencia (Sollins, 1992)

DRBD es un módulo ubicado en el kernel, este módulo crea un dispositivo de bloque virtual vinculado a un dispositivo de bloque físico, el dispositivo de bloque virtual es usado en lugar del dispositivo de bloque físico, esto permite que toda operación de escritura y lectura enviada a este dispositivo de bloque pueda ser manipulada por el controlador DRBD, agregando de forma transparente alta disponibilidad y protección de los datos para toda aplicación.

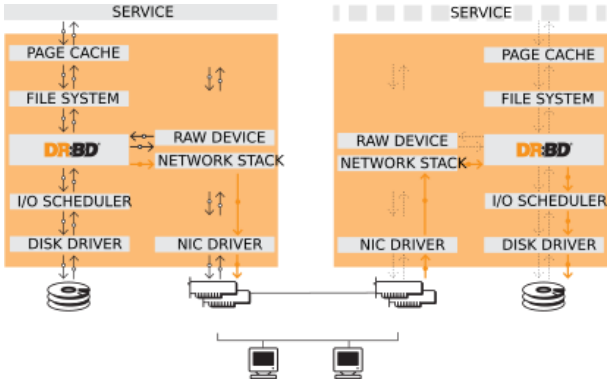


Figura (05) - DRBD

II.2.2. Network Block Device (NBD)

Este servicio convierte cualquier servidor remoto como un dispositivo de bloques local, es decir que habiendo dos ordenadores A y B, toma de A un archivo y lo mapea en B como una unidad de disco, escribiendo y leyendo en A lo que se escriba y lea en la unidad mapeada en B. Esto permite agregar en un ordenador A, el almacenamiento que está disponible en un ordenador B.

NBD es ejecutado por los siguientes componentes:

- ✓ El servidor
- ✓ El cliente
- ✓ La red entre ellos

En el cliente un controlador a nivel del kernel controla toda actividad que se realice en el dispositivo, cuando un programa intenta manipular el dispositivo, el kernel realiza un reenvío de la petición al servidor en el que los datos residen

físicamente. Todas las peticiones recibidas por el servidor son manipuladas por un programa de espacio de usuario.

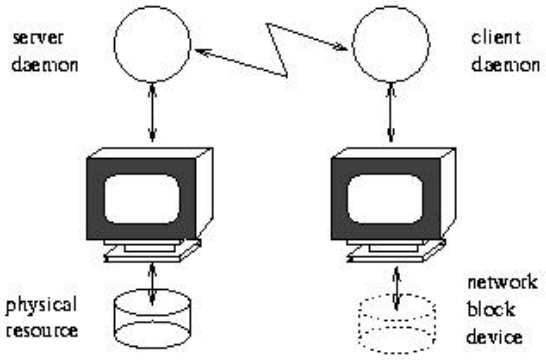


Figura (06) - NBD

II.2.3. Tecnologías para respaldo, seguridad y alta disponibilidad

El respaldo de la información consiste en realizar copias de los datos generados por una aplicación ubicada en un dispositivo primario, hacia uno o más dispositivos secundarios, con el propósito de resguardarlos ante posibles fallos de la fuente primaria que pudieran ocasionar la pérdida de éstos.

Existen diferentes tipos de respaldos los cuales se mencionan a continuación:

II.2.4. Respaldos por Procedimiento

II.2.4.1. Respaldo Manual

Es un respaldo realizado directamente por una persona ya sea a voluntad propia o previamente calendarizado para una fecha y hora determinada. Este respaldo puede llevarse a cabo, según el sistema operativo que se esté usando, mediante el uso de comandos o por medio de alguna aplicación de exploración de archivos.

II.2.4.2. Respaldo Automático

Es un respaldo que se realiza por medio de una aplicación especializada, configurada previamente por una persona, indicando los recursos a ser respaldos

con su debida calendarización, permitiendo así un respaldo en tiempo real. Este respaldo puede ser configurado una única vez y ser ejecutado tantas veces sea requerido.

Si bien es cierto que los respaldos automáticos una vez configurados, no necesitan más interacción de una persona para ejecutarse, siempre es necesario que posterior al proceso de respaldo alguien verifique que este haya sido ejecutado satisfactoriamente y que los datos respaldados no se hayan corrompido y mantenga su integridad.

Existen diferentes tipos de respaldos automáticos los cuales se mencionan a continuación:

II.2.4.2.1. Respaldo Lineal

Este tipo de respaldo consiste en que cada transacción o modificación de datos que se realice será respalda inmediatamente. Se sacrifica el performance o desempeño a cambio de un respaldo instantáneo.

II.2.4.2.2. Respaldo Diferido

Este tipo de respaldo, a diferencia del lineal, se realiza a una hora determinada, generalmente cuando las transacciones más importantes hayan sido realizadas y que la carga del sistema sea muy pequeña, por ejemplo, a medianoche o durante la madrugada.

II.2.5. Respaldos por alcances y cobertura de datos

II.2.5.1. Respaldos Completos

Estos respaldos son una copia completa de los datos originales, por lo que el tamaño del respaldo aun compreso tendrá un tamaño cercano al de los datos originales, es importante mencionar que, en este tipo de respaldos, también se respaldan datos innecesarios.

II.2.5.2. Respaldos Incrementales

Estos respaldos copian solamente los datos nuevos a partir del último respaldo realizado, en caso de no existir un respaldo previo o por ser la primera vez se respaldaría toda la información.

Este tipo de respaldo se puede realizar de dos formas:

II.2.5.2.1. Respaldo Incremental Asíncrono

Este tipo de respaldo solo copia datos nuevos sin preocuparse de la información faltante o de la duplicada.

II.2.5.2.2. Respaldo Incremental Síncrono

En este tipo de respaldo se agregan los datos nuevos existentes solo en el origen, los datos idénticos en el origen y el respaldo se preservan, y los datos que solo existen en el respaldo, pero no en el origen son eliminados.

II.2.5.3. Respaldos Versionados

Se respaldan instantáneas de todo el repositorio en cada versión, o instantáneas de las diferentes versiones de cada elemento de información.

En este tipo de respaldo se pueden incluir fechas, horas o cualquier otro método para controlar la versión del respaldo.

II.3. Almacenamiento de datos

El respaldo puede almacenarse en dispositivos de bloques locales como Hard Drives, USB, CD/DVD o en almacenamientos remotos.

Los datos deben de almacenarse hasta que sea necesario, ya sea por uno, o más métodos de almacenamiento. Existen distintos tipos de almacenamiento y existe una interrelación entre costo, velocidad y capacidad.

Los medios de almacenamiento de alta velocidad cuestan más por byte y proporcionan capacidades menores. Los medios de almacenamiento de gran capacidad cuestan menos por byte, pero son más lentos.

Entre las deficiencias del almacenamiento mediante dispositivos de bloques locales, producen como problema la inconsistencia de los datos, cuando existen múltiples fuentes y destinos, incluso el problema de perderlos o no encontrarlos al tener tantos respaldos físicos, estas deficiencias y problemas son muy conocidas, sin embargo, bien podría ser que la empresa tome la decisión de seguir lidiando con estas deficiencias en lugar de migrar a un nuevo método de almacenamiento.

Para el almacenamiento de datos remotos el servicio más usual es el NFS.

II.3.1. Network File System (NFS)

Es un protocolo del modelo OSI a nivel de aplicación, que sirve para implementar sistemas de archivos distribuidos, permitiendo que varios ordenadores tengan disponibilidad y acceso a los datos mediante la red.

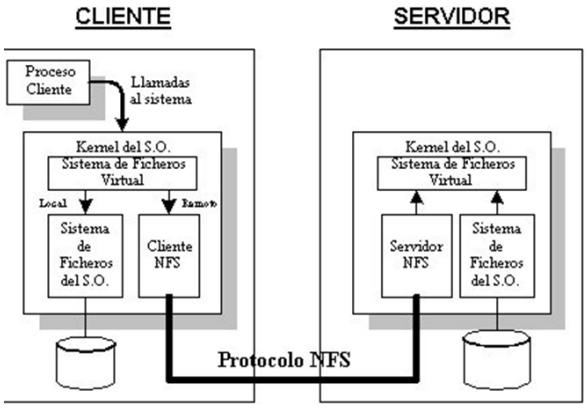


Figura (07) - NFS

II.3.2. Virtual Private Network (VPN[19])

Es un término genérico que cubre el uso de redes públicas o privadas entre un grupo de usuarios que están separados de otro grupo de usuarios en otra red, y crean una red privada permitiéndoles comunicarse entre ellos.

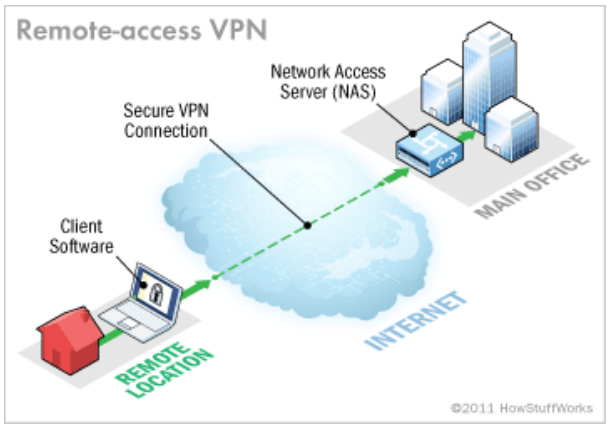


Figura (08) - VPN

II.3.3. Heartbeat

Este servicio o "demonio" de sistemas operativos derivados de UNIX se encarga de enviar señales de monitoreo para validar si algo está o no funcionando correctamente.

Este demonio sirve de base para crear una infraestructura de servidores que funcionen como un clúster de alta disponibilidad[20], permitiendo mediante el intercambio de mensajes entre sus miembros, el enterarse de la presencia o

---

[19] **VPN (Virtual Private Network):** es una tecnología de red que se utiliza para conectar una o más computadoras a una red privada utilizando Internet. Ver detalles en referencia (Gleeson, Lin, Heinanen, Armitage, & Malis, 2000)

[20] **Alta Disponibilidad:** es un protocolo de diseño del sistema y su implementación garantiza la continuidad o disponibilidad operacional de los servicios, incluso en situaciones de deficiencias de hardware, software, corte de energía, etc. (Ver sección II.8).

ausencia de cada uno de ellos, así como el estado de los servicios que se ejecutan en cada uno de ellos.

Al momento de un fallo, Heartbeat se encarga de realizar las operaciones correspondientes para su manejo y recuperación.

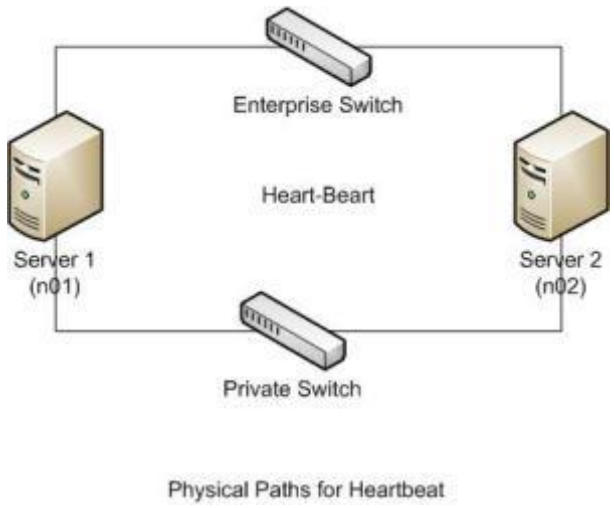


Figura (09) - HEARTBEAT

II.4. Redundancia en red: LACP

LACP (Link Aggregation Control Protocol) es también conocido como "BONDING", es una técnica usada para vincular muchas interfaces de red conectados a distintos dispositivos de red, o en ocasiones a puertos distintos de un mismo dispositivo de red para crear el grupo como un solo enlace o dispositivo de red. Con esta técnica es posible lograr distintos objetivos, tales como:

- **Tolerancia a fallos.**
- **Aumento de rendimiento.**
- **Todas las anteriores.**

El factor clave para esta técnica es que permite la alta redundancia, protege el ambiente de servidores ante la pérdida o el fallo de servicios ante fallos físicos de dispositivos de red.

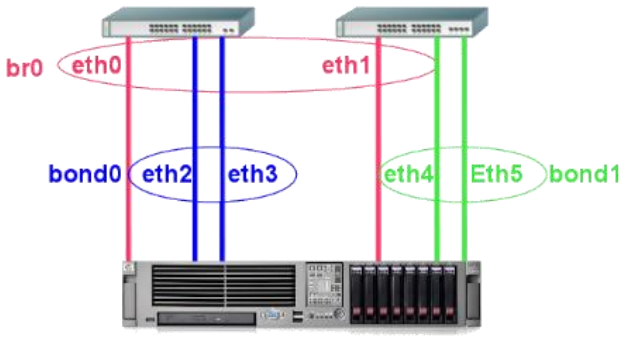


Figura (10) - Bonding o LACP

Esta técnica permite tres modos de funcionamiento:

1. **Activo/Pasivo:** existe solo una interfaz de red activa, en caso que esta falle, una de las que están en estado pasivo toma su lugar.
2. **Agregado de enlace:** todas las interfaces de red funcionan como un solo dispositivo para tener un mayor rendimiento.
3. **Balance de carga:** el tráfico de red es balanceado equitativamente a través de las interfaces de red miembros del bonding.

II.5. Redundancia en el almacenamiento: RAID

El RAID (Redundant Array of Independent Disks) o Arreglo Redundante de Discos Independientes, es un sistema de almacenamiento de datos que usa un conjunto de discos independientes organizados en uno o más niveles, con la finalidad de actuar como un solo disco, esta tecnología ofrece uno o más de los siguientes beneficios: mayor integridad, mayor tolerancia a fallos, mayor throughput (rendimiento o productividad) y mayor capacidad.

Las distintas configuraciones o niveles de RAID son las siguientes[21]:

[21] **Las distintas configuraciones o niveles de RAID** fueron tomadas de referencia de la siguiente cita: (Common RAID Disk Data Format Specification, 2019)

RAID0: esta configuración permite tener un conjunto de discos unificados creando un solo volumen. Su propósito es distribuir los datos en todos los miembros del arreglo para incrementar el tamaño del volumen. Ofrece un alto rendimiento debido a la escritura y lectura realizada de forma paralela. En esta configuración no se favorece tanto la redundancia como la capacidad, pues basta que falle un disco en el arreglo, para que se pierdan los datos, puesto que éstos están distribuidos en todo el arreglo.

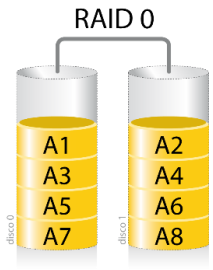


Figura (11) - RAID0

RAID1: esta configuración permite crear una copia exacta de un conjunto de datos en dos o más discos. Esta configuración se le conoce como "espejo", es un arreglo fiable, puesto que al momento de un fallo los datos están presentes en el volumen copia. Todo dato que se escribe se duplica en todos los discos, permitiendo la seguridad de los datos y su persistencia ante un fallo.

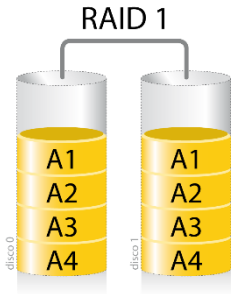


Figura (12) - RAID1

RAID3: es una configuración con método de división a nivel de bytes. Permite configurar un conjunto de discos creando un volumen seccionado, con paridad no rotativa, está optimizado para transferencias sencillas y largas, como son el

streaming de video, gráficos y ediciones de video, este arreglo es pésimo con solicitudes de datos pequeños.

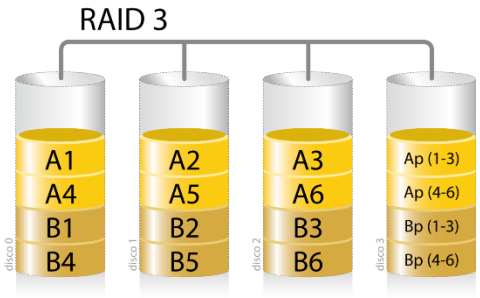


Figura (13) - RAID3

RAID4: es una configuración con método de división a nivel de bloques. Permite configurar un conjunto de discos creando un volumen seccionado, con paridad no rotativa, está optimizado para transferencias múltiples y cortas, especialmente si es de lectura exclusivamente, por ejemplo, para sistemas de correos.

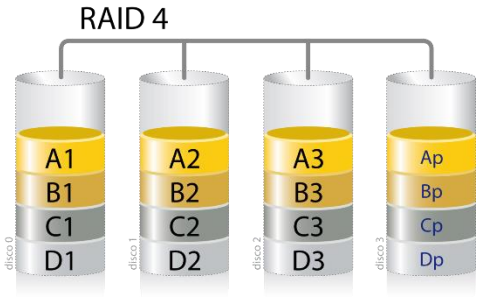


Figura (14) - RAID4

RAID5: es una configuración con método de división a nivel de bloques. Permite configurar un conjunto de discos creando un volumen seccionado, con paridad rotativa, está optimizado para transferencias múltiples y cortas, es recomendable para sistemas multiusuario para rendimiento no crítico y con pocas operaciones de lectura.

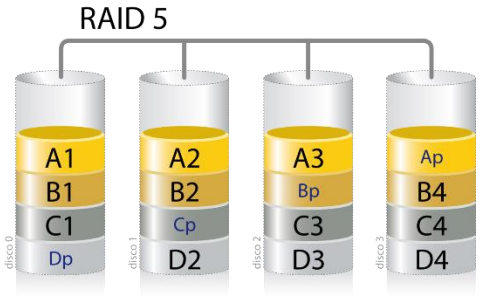


Figura (15) - RAID5

RAID6: es una configuración similar a RAID5 con la diferencia que la paridad rotativa es dual, es tolerante al fallo de hasta dos discos.

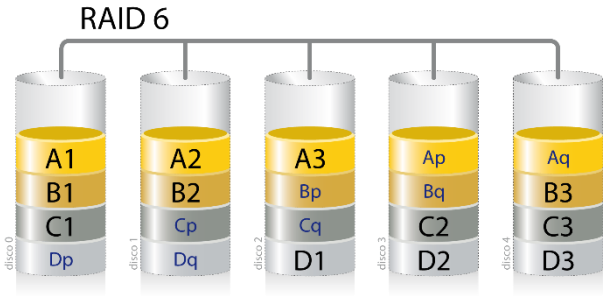


Figura (16) - RAID6

RAID10: es una configuración combinada de **RAID1+RAID0**, en cuanto al proceso de duplicación, es rápido y a prueba de fallos, pero no está salvo de corrupción y daño de datos, como lo es un RAID que utiliza paridad.

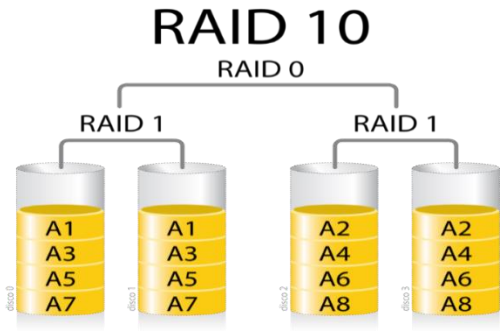


Figura (17) - RAID10

MDF: es una configuración similar a RAID6, pero soporta el fallo de más de dos discos.

RAID-1E: esta configuración es similar al RAID10, pero integra el seccionado de volúmenes en el arreglo.

RAID-5E: esta configuración es similar al RAID5, pero agrega al final un espacio de reserva al final de cada extensión.

RAID-5EE: esta configuración es similar al RAID5, pero agrega un espacio de reserva dentro de cada extensión.

RAID-5R: esta configuración es similar al RAID5, con paridad rotativa luego de la configuración de un número de volúmenes seccionados.

Los RAID pueden según su configuración mejorar la disponibilidad de los datos y en ocasiones el rendimiento de los dispositivos, sin embargo, no protege 100% los datos y en caso de un fallo total la pérdida de los datos es irreversible.

II.6. Clúster[22]

El grupo de ordenadores que trabajan en conjunto se define como Clúster, su funcionalidad es que el grupo funcione como uno solo servidor. Un clúster puede trabajar en dos modos:

1. Alto desempeño
2. Alta disponibilidad

[22] **Clúster:** Ver detalles en las siguientes referencias: (Paul Kirvan, 2006), y (Cluster, 2015).

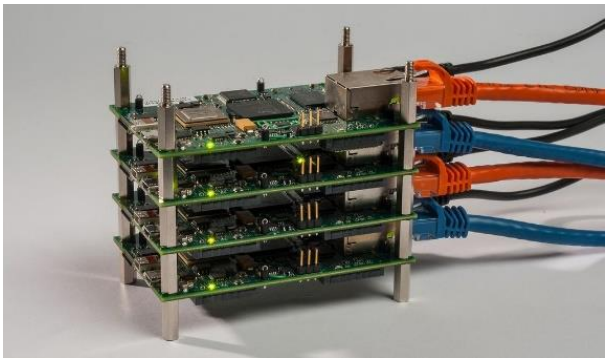


Figura (18) - Clúster

Para ofrecer alto desempeño o alta disponibilidad, el clúster debe controlar cada uno de sus integrantes y sus componentes mediante frameworks[23] que monitorean el hardware, el software y la capa de transporte[24]. La capa de transporte es la cuarta capa en el modelo OSI (ISO/IEC 7498-1), cuyo propósito es de proveer una transferencia de datos transparente entre distintas entidades de sesión y liberarlos de cualquier detalle que pueda provocar un alto costo en su entrega confiable de datos para que los datos o los recursos estén siempre disponibles.

II.7. Servidores esclavos y maestros

Todo ordenador miembro de un clúster puede funcionar como maestro o esclavo, pero no ambos al mismo tiempo.

---

[23] **Frameworks:** en el desarrollo de software, un entorno de trabajo es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software. Ver detalles en referencia (Lutkevich, 2015)

[24]**. Capa de transporte:** es el cuarto nivel en el modelo de interconexión de sistema abierto (OSI), y está encargado de la transferencia libre de errores de los datos entre el emisor y el receptor. Ver detalles en referencia (ISO/IEC 7498-1, 1996)

Por lo general se instalan los mismos servicios en ambos servidores, sin embargo, esto no es obligatorio, ya que depende de lo que se quiera tener configurado con una finalidad en específico.

La diferencia entre los servidores maestros y esclavos radica en que el maestro es el que tiene el control total y posee los servicios en ejecución, mientras que el servidor esclavo posee servicios detenidos, a la espera de un fallo del servidor maestro, de esta forma el servidor esclavo tomaría el control al momento del fallo, asegurando así la disponibilidad de los servicios.

Es menester definir la modalidad en que se ejecutará cada nodo virtual (la modalidad puede ser maestro o esclavo) existente en cada nodo físico, puesto que se requiere HA (High Availability/Alta Disponibilidad) dentro del clúster.

A continuación, se enumeran las reglas que definen la funcionalidad de cada nodo virtual dentro del clúster para sus estados maestros y esclavo:

a. Debe haber un Nodo Virtual Maestro en cada nodo físico.
b. Debe haber un Nodo Virtual Esclavo en cada nodo físico.
c. Las reglas anteriores definen que se requiere dos nodos virtuales por cada nodo físico.
d. Cada nodo virtual esclavo debe de ser respaldo de un maestro en un nodo físico distinto al que se encuentra el nodo virtual maestro.
e. Solo al momento de un fallo que deje fuera de servicio un nodo físico y a sus dos nodos virtuales, uno de los nodos físicos miembros del clúster tendrá en ese momento dos nodos virtuales maestros. Al momento de restaurarse el nodo físico que falló y estar en línea nuevamente, uno de sus nodos virtuales asumirá el control de maestro y posteriormente este enviará una orden al nodo virtual que le suplantaba como maestro para que vuelva a su condición inicial de esclavo, quedando de esta manera en cada nodo físico un nodo virtual maestro y un nodo virtual esclavo.

II.8. Alta Disponibilidad[25]

Un clúster de alta disponibilidad es un conjunto de ordenadores que trabajan síncrona o asíncronamente para mantener un mismo servicio en común siempre disponible.

El objetivo de la alta disponibilidad es crear un sistema tolerante a fallos, este sistema permite detectar el fallo y caída de uno de los miembros del clúster y tomar las medidas y acciones necesarias para levantar uno de los miembros del arreglo para dar continuidad del servicio haciendo que los datos estén siempre disponibles mediante la réplica de estos y la disposición inmediata de los recursos.

La orientación de la alta disponibilidad apunta a la disponibilidad permanente, alta redundancia e integridad de los datos. Esto requiere que el servicio tenga dos capas, una que se encargue de proteger los fallos de hardware y otra capa que se encargue de los fallos de software.

Todo clúster de alta disponibilidad debe de enfrentarse a la toma de decisiones en cuanto a la acción a tomar al momento de un fallo determinado. Cuando se produce un fallo inician ciertas problemáticas de las cuales el clúster debe hacerse cargo para evitar la pérdida de la integridad de los datos, las decisiones a tomar en cada estado del clúster son las siguientes:

a. **FailOver**[26]**:** se da cuando el miembro del clúster que tiene control de los servicios falla, al cual se le etiqueta como servidor maestro, el clúster decide a cuál servidor esclavo se le atribuye la entrega del control de estos servicios.

[25] **Alta Disponibilidad:** Ver detalles en referencia (LINBIT HA-Solutions GmbH [Linux-HA], 2010), (Paredes, 2001)

[26] **FailOver:** ver detalles en referencia (Failover, 2015)

b. **TakeOver**[27]**:** se da cuando un miembro del clúster sale de su estado de fallo y se reintegra al clúster, se debe de tomar una decisión, si este servidor era maestro y debe volver a serlo, o solo es integrado como un esclavo más.
c. **SwitchOver**[28]**:** se da cuando un miembro del clúster cede los recursos a otro miembro del clúster bajo una condición automática programable o por una acción manual, provoca que un miembro del clúster que posee el control maestro pase a ser esclavo y un miembro esclavo del clúster pase a ser maestro**.**
d. **SplitBrain**[29]**:** se da cuando la comunicación entre uno o más miembros del clúster se pierde, sin embargo, el estado de los miembros es saludable, por consiguiente, todos toman el rol de maestro; es necesario mencionar que en el clúster solo un servidor puede tomar el rol de maestro o cerebro (brain).

El término SplitBrain proviene de la actividad que sucede en ese momento, debido que al restablecerse la comunicación entre los miembros, todos estos vuelven a comunicarse bajo el rol de maestro, dándose un SplitBrain (es decir cerebro divido), ya que uno o más miembros tienen el rol de maestro compitiendo a la vez por el control sin soltarlo, al no haber quorum, el clúster nunca reanuda su estado, un ejemplo probable para este tipo de fallo es que se dé un cuello de botella, o saturación entre la comunicación de los miembros. El problema grave de esta situación son los datos y su integridad, el miembro maestro es el encargado del arbitraje para los datos, y al haber más de un miembro con el rol maestro, el arbitraje para los datos se corrompen, y no pueden dialogar los miembros del clúster; para evitar esto, se deben de crear scripts, o configurar banderas de control que consulten el estado de cada miembro, y al momento de que el resultado da

[27] **TakeOver:** ver detalles en referencia (TakeOver, s.f.)

[28] **SwitchOver:** ver detalles en referencia (SwitchOver, s.f.)

[29] **SplitBrain:** ver detalles en referencia (HLINBIT HA-Solutions GmbH [SplitBrain], 2010)

positivo a un estado de SplitBrain, se indica una señal al conjunto de ordenadores miembros para ponerse de acuerdo de la acción a tomar, y cuál de todos tomará el control.

II.9. Alto Desempeño[30]

El alto desempeño por el contrario de la alta disponibilidad no se centra en los datos, más bien se enfoca en los recursos de hardware que posee el clúster.

El punto de partida fundamental para desarrollar un clúster de alto desempeño es la necesidad de recursos de hardware suficientes para resolver problemas muy complejos, permitiendo realizar cálculos matemáticos y simulaciones en el menor tiempo posible.

La capacidad del desempeño del clúster es medida en FLOPS[31] (acrónimo de **Floating-point Operations Per Second / Operaciones de coma flotante por segundo**) la cual es inversamente proporcional al tiempo de ejecución de las tareas, es decir que a mayor capacidad del clúster menor tiempo tomará en realizar dichas tareas y viceversa.

Hay que tener en cuenta que el clúster se verá limitado por el ordenador con menor capacidad en sus recursos de hardware, debido a que el clúster está compuesto por varios ordenadores trabajando en conjunto para realizar una tarea, es por ello que si uno de los equipos posee un hardware inferior, hará que el resto

---

[30] **Alto Desempeño:** el campo en la informática del alto rendimiento (High performance Computing o HPC en inglés) consiste en un conjunto de tecnologías computacionales que permiten obtener alto rendimiento de cómputo. Ver detalles en referencias (Arjuna, 2013) y (Softpanorama, s.f.)

[31] **FLOPS:** son una medida de rendimiento de una computadora, especialmente en el campo científico, en donde se utiliza mucho las operaciones con datos de tipo flotante, para realizar simulaciones precisas y obtener resultados fidedignos. Ver más información en referencias: (TechTerms, 2009), (Computer Hope, 2018), (Webopedia, 1998)

de miembros tenga que esperar que termine de realizar su tarea para que el clúster pueda continuar con la siguiente, por esto no es recomendable la creación de clúster de alto desempeño con equipos que no posean requerimientos técnicos similares.

Cada ordenador posee recursos de hardware que van desde el procesador, la memoria y otros recursos como video y disco duro. Hoy en día los ordenadores pueden poseer más de un procesador, así como memorias de alta frecuencia y de alta capacidad. Estos equipos y recursos actualmente son accesibles en cuanto al costo, permitiendo que los miembros del clúster y sus recursos puedan ser comprados en cualquier tienda, para luego ser armados a un precio razonable.

Cuando una empresa desea crear un clúster de alto desempeño lo primero que hace es definir los requerimientos técnicos de los equipos que conformarán dicho clúster para resolver su necesidad (Memoria RAM[32], CPU[33], GPU[34], etc.) para luego cotizar en el mercado el precio del hardware previamente definido. Como se indicó anteriormente se recomienda buscar equipos con recursos idénticos o similares para que el clúster tenga un desempeño óptimo aprovechando al máximo dichos recursos. Cuando se tienen equipos demasiado rápidos dentro de un clúster ya existente con equipos inferiores, y variables en un estado permanente es hacer una mala inversión, ya que el clúster funcionará a la velocidad de los equipos más lento.

---

[32] **RAM:** (Random Access Memory - Memoria de Acceso Aleatorio), memoria principal de acceso aleatorio de la computadora, donde residen programas y datos, sobre la que se pueden efectuar operaciones de lectura y escritura.

[33] **CPU:** (Central Process Unit – Unidad Central de Procesamiento), es el recurso de todo ordenador donde se llevan a cabo cada procesamiento de todas las instrucciones mediante las operaciones básica aritméticas, lógicas y de Entrada y Salida del sistema.

[34] **GPU:** (Graphic Process Unit – Unidad de Procesamiento Gráfico), es el recurso de todo ordenador dedicado para el procesamiento de todas las instrucciones de coma flotante, para quitarle carga al CPU y entregar al GPU otro tipo de cálculos liberando así tareas y mejorando el desempeño del ordenador.

Estos escenarios solo deberían de estar presentes cuando lo que se va a realizar es un reemplazo gradual del hardware existente por uno más moderno.

II.10. Dispositivos de Bloques en Red

Un dispositivo de bloque[35] es un periférico de almacenamiento que está conformado por bloques de tamaño específico que pueden tener un tamaño usual de entre 512 bytes a 65,536 bytes.

Hablar de dispositivos de bloques en red[36], es hablar de un almacenamiento de información en bloques de tamaño fijo con su propia dirección, donde estos bloques no se encuentran localizados físicamente en el ordenador local, sino que son transferidos por algún protocolo en red de forma remota.

II.11. Virtualización

La virtualización es una técnica que está basada en la abstracción de los recursos de una computadora, llamada *Hypervisor* o VMM (*Virtual Machine Monitor*) que crea una capa de separación entre el hardware de la máquina física (*host*) y el sistema operativo de la máquina virtual (*virtual machine, guest*), y es un medio para crear una "versión virtual" de un dispositivo o recurso, como un servidor, un dispositivo de almacenamiento, una red o incluso un sistema operativo, donde se divide el recurso en uno o más entornos de ejecución (Suppi Boldrito, 2015), (Brush & Kirsch, 2010).

[35] **Dispositivo de Bloque:** ver detalles en referencia (Tanenbaum, Dispositivos de E/S, 2009)

[36] **Dispositivos de Bloques en Red:** ver sección II.10 para ver los servicios utilizados para replicar o publicar bloques en la red.

Figura (19) - Virtualización

Esta técnica saca el máximo provecho de un ordenador. De acuerdo con la técnica de virtualización usada, el software puede darse cuenta o no, que el hardware en el que se está ejecutando es ficticio o creer que es real, para ello el software virtualizador emula el procesamiento del hardware físico. El hardware emulado al nivel del software devuelve la misma información de un hardware real, por lo tanto, cada entorno virtualizado puede ser aislado y tener un fork[37] independiente. Cada entorno requiere recursos que son particionados y pueden ser modificados (en tiempo de ejecución si el virtualizador lo permite) permitiendo escalabilidad y balancear cargas de trabajo en el hardware anfitrión.

El software que administra la virtualización puede hacer uso de una de las siguientes técnicas de virtualización:

II.11.1. Virtualización completa

Emulación de hardware ficticio o real, tomando recursos reales del anfitrión, capa responsable de la administración de los recursos que toma el nombre de hipervisor, permitiendo crear plataformas emuladas y poder correr arbitrariamente distintos

[37] **Fork:** una bifurcación o fork, cuando se aplica en el contexto de un lenguaje de programación o un sistema operativo, hace referencia a la creación de una copia de sí mismo por parte de un programa, que entonces actúa como un "proceso hijo" del proceso originario, ahora llamado "padre". Los procesos resultantes son idénticos, salvo que tienen distinto número de proceso (PID). Ver detalles en referencia (Tanenbaum, Fork, 2009)

sistemas operativos sin modificaciones ya que el sistema operativo no se da cuenta que está corriendo en un hardware ficticio.

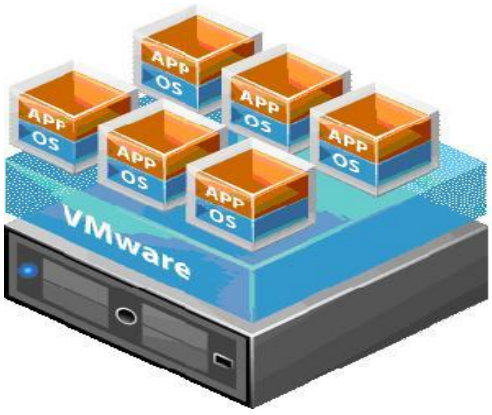


Figura (20) - Virtualización completa

Existen dos tipos de hipervisores dentro de esta categoría, los Hipervisores de tipo 1 conocidos como hipervisor nativo, y los de tipo 2 conocidos como hipervisor hospedado.

II.11.1.1. Hipervisor tipo 1

Este hipervisor se instala directamente en el kernel, en lo más adyacente al hardware físico y no depende de un sistema operativo para funcionar, por lo que todos los controladores están contenidos por el mismo hipervisor, dando alto rendimiento y estabilidad. Esta virtualización es adecuada para virtualización puesta en producción.

Como hipervisor de tipo 1 tenemos: VMWare ESXI (VMWARE ESXI, 2016), KVM.

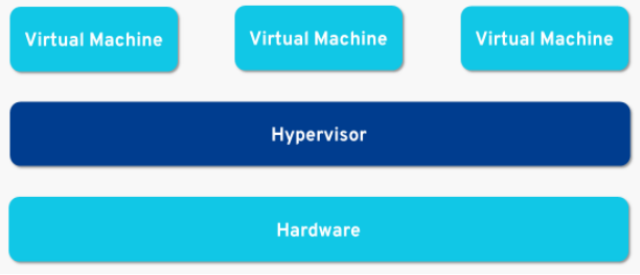


Figura (21) - Hipervisor tipo 1

II.11.1.2. Hipervisor tipo 2

Este tipo de hipervisor requiere ser instalado sobre un sistema operativo como un programa más. Se instala como cualquier otro programa. El hipervisor maneja y gestiona la virtualización, no necesita tener los controladores ya que de esto se encargaría el Sistema Operativo de transferir todas las solicitudes del software hacia el hardware, resultando ser un hipervisor de bajo rendimiento y de pobre desempeño. Este tipo de hipervisor es adecuado para el desarrollo antes de puesta en producción.

Como hipervisor de tipo 2 tenemos: VMWare Workstation, Virtual Box (Virtual Box, 2016), QEMU, Parallels, Microsoft Virtual Server.

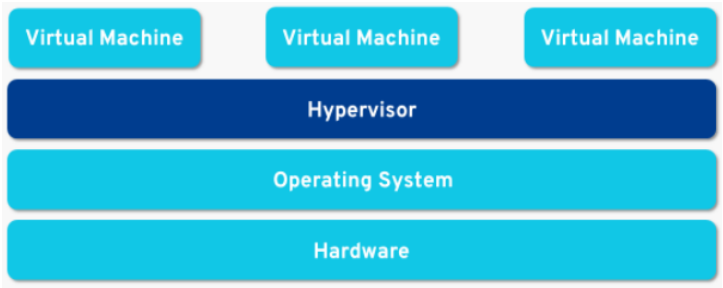


Figura (22) - Hipervisor tipo 2

II.11.2. Paravirtualización

La tecnología de virtualización permite que la mayoría del trabajo sea realizado en el Sistema Virtual, el cual está hecho para soportar esta parcialidad y el resto de código es ejecutado en el Anfitrión. La paravirtualización permite tener arbitrariamente distintos sistemas operativos con la excepción que el Sistema Operativo debe de saber que ésta funcionando de forma virtual. Esta metodología es usada por (XEN, 2016) y Unified Modeling Language (UML).

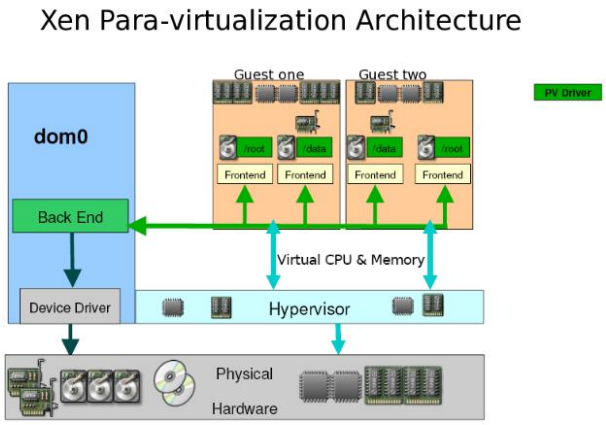


Figura (23) - Paravirtualización

II.11.3. Por contenedores

Esta técnica busca compartir la misma máquina con otros aislando cada aplicación de forma segura en un ambiente enjaulado. Esta técnica no permite tener distintos Sistemas Operativos, sino más bien múltiples instancias, esto permite tener distintas distribuciones o copias del propio Sistema Operativos de forma aislada: Ejemplos de virtualización por contenedor son: (Virtuozzo, 2016), (Linux-VServer, 2013), Solaris Zones and (Fundación FreeBSD, 2016) Jails.

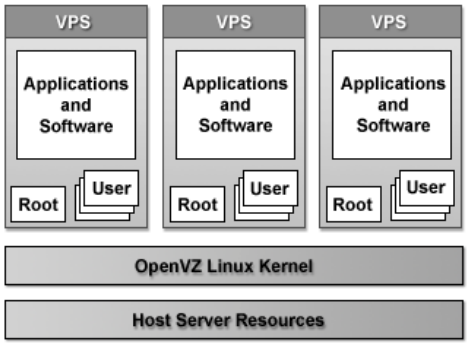


Figura (24) - Contenedores

CAPITULO III. DISEÑO DEL SISTEMA DE RESPALDO EN LÍNEA BASADO EN RAVD

III.1. Especificación del Sistema de Respaldo

III.1.1. Modelo conceptual del sistema de almacenamiento de Arreglo Redundante de Discos Virtuales (RAVD)

Conceptualmente, el Sistema de Respaldo en Línea basado en RAVD es un sistema de almacenamiento en red implementado sobre un Arreglo Redundante de Discos Virtuales (RAVD). Es una solución similar a NAS, o a un servicio de almacenamiento en nube, en los cuales, los clientes pueden acceder a archivos o contenido digital alojado en el sistema a través de la red local en el caso de NAS, o el Internet en el caso de la nube, mediante aplicaciones de acceso a los correspondientes servicios, mediante protocolos de red (como, por ejemplo, NFS, SMB, FTP o streaming sobre HTTP).

La diferencia radical con éstos es cómo se implementa. El modelo RAVD se implementa sobre un clúster de servidores. Este clúster se compone de dos subclústers interdependientes entre sí. El primero conforma el espacio de almacenamiento (storage) y se conforma por ordenadores denominados “nodos S” o SPN (siglas de “Storage Physical Node”). El segundo conforma el controlador de arreglo de discos que sirve el clúster de almacenamiento y se compone de ordenadores denominados “nodos C” o CPN (siglas de “Controller Physical Node”).

Los nodos S proveen servicio de almacenamiento mediante discos virtuales a los nodos C (los cuales son accedidos de forma remota). Los nodos C juegan el papel de controlador de arreglo de discos redundantes (en forma análoga a una controladora RAID). Asimismo, el nodo C suministra el servicio de almacenamiento a los clientes finales mediante unidades o volúmenes lógicos que son accedidos a través de los servicios de acceso antes mencionados.

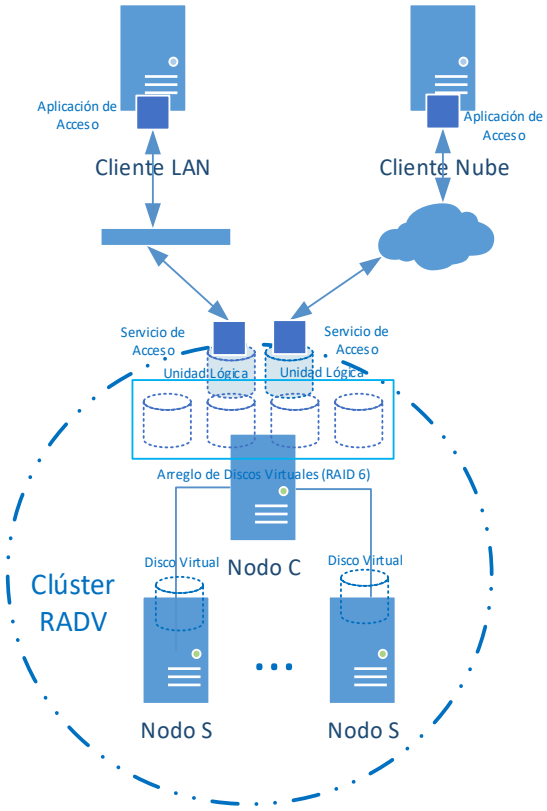


Figura (25) - Modelo Conceptual del Sistema de Almacenamiento en línea basado en RAVD

El clúster RADV, fue diseñado para proveer alta fiabilidad de datos, alta disponibilidad y alto rendimiento.

III.1.2. Arquitectura del Clúster RAVD

Como todo clúster, **RAVD** está conformado por una cantidad **N** de ordenadores físicos o nodos que trabajan en conjunto como una unidad, para suministrar un servicio (almacenamiento en red este caso).

Debido a la separación de responsabilidades para proveer la función del clúster (suministrar servicio de almacenamiento en línea), éste se conforma por dos subclústers interdependientes: el **subclúster de almacenamiento** (compuesto por nodos S o SPN) y el **subclúster de control** (conformado por nodos C o CPN).

III.1.2.1. Estructura del subclúster de almacenamiento

El subclúster de almacenamiento del clúster RAVD se conforma por **S** "nodos S" que son ordenadores físicos, denominados abreviadamente **SPN (Storage Physical Node)**. Para proveer alta disponibilidad, cada nodo del subclúster implementa redundancia con su nodo par homólogo, en modalidad maestro-esclavo, con replicación automática, en línea, y en tiempo real del disco virtual. Para ello, el **SPN** contiene dos instancias de servidor virtual[38] denominados **SVN (Storage Virtual Node)**. Uno de estos **SVN** está configurado como maestro: **MA-SVN (Master Storage Virtual Node)** y el otro, como esclavo: **SL-SVN (Slave Storage Virtual Node)**.

Para implementar la alta disponibilidad, cada **MA-SVN** tiene su correspondiente **SL-SVN**, el cual **está ubicado en un SPN distinto**. Si un **MA-SVN** falla, será reemplazado por su **SL-SVN** correspondiente en otro **SPN**, el cual es promovido a master inmediatamente que se detecta el fallo.

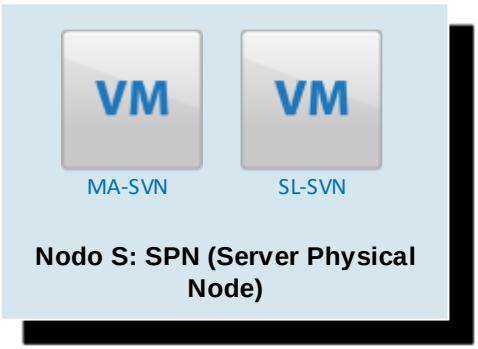


Figura (26) - Estructura de un nodo físico S del Clúster RAVD: Server Physical Node (SPN)

Se designa por "**S**" a la cantidad total de nodos **SPN** del subclúster de almacenamiento. Nominalmente, S debe ser siempre mayor o igual a cuatro por

[38] Cada servidor virtual se implementa sobre una máquina virtual alojada en un software de virtualización o hipervisor que se ejecuta en el servidor físico SPN.

cuanto la configuración del arreglo definida para el RAVD es RAID6, el cual requiere al menos cuatro discos y cada **SPN** sirve un disco virtual.

En vista de la posibilidad de tener discos activos en el arreglo y discos de repuesto (spare disks), se designa como $\mathbf{S_a}$ a la cantidad total de nodos SPN que sirven discos activos y $\mathbf{S_s}$ a la cantidad total de nodos SPN que sirven discos spare. De forma tal que $\mathbf{S = S_a + S_s}$.

III.1.2.2. Estructura del subclúster de control

El subclúster de control se conforma por **C** "nodos C" que son ordenadores físicos, denominados abreviadamente **CPN (Controller Physical Node)**. Cada **CPN** contiene una instancia de servidor virtual o **CVN**. Para proveer alta disponibilidad, el subclúster es redundante mediante dos "nodos C" en modelo maestro-esclavo, implementados sobre dos **CPN** diferentes. El configurado como maestro, tendrá un **MA-CVN (Master Controller Virtual Node)** y el configurado como esclavo tendrá un **SL-CVN (Slave Controller Virtual Node)**. Solo uno de ellos (el maestro) estará activo a la vez.

La función del **MA-CVN** será gestionar el arreglo de discos virtuales y dar acceso a los clientes finales a los recursos del almacenamiento a través de unidades o volúmenes lógicos.

El **SL-CVN** queda como respaldo, y se activará solamente si es promovido a maestro, en caso de que el **MA-CVN** falle.

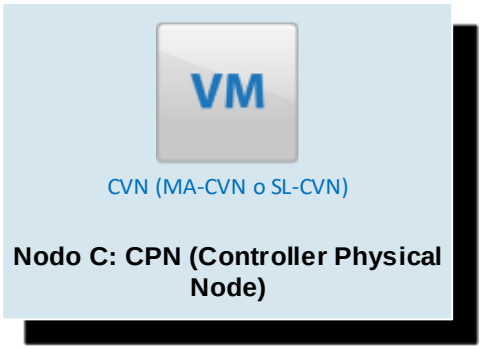


Figura (27) - Estructura de un nodo físico C del Clúster RAVD: Controller Physical Node (CPN)

El arreglo de discos se configura en **MA-CVN** a partir de los discos virtuales servidos por el subclúster de almacenamiento. Para el modelo **RADV** se ha definido que la configuración del arreglo de discos más adecuada es **RAID6** por ser la que mejor cumple con los requisitos de alta fiabilidad con la menor cantidad de discos, soportando caída de hasta dos discos virtuales del arreglo simultáneamente sin pérdida de datos. RAID6 requiere para su funcionamiento al menos cuatro discos.

Algunos de los discos virtuales se pueden reservar como repuesto en línea (spare disks). Éstos no conforman parte del arreglo, pero están en espera para proveer reposición automática en línea de los discos activos del arreglo ante una eventualidad de falla.

Se designa por "**C**" a la cantidad total de nodos **CPN** del subclúster de control. Conforme la definición arquitectural del RAVD, **C es constante e igual a dos**.

III.1.2.3. Estructura del clúster RAVD

El clúster RAVD se conforma por los dos subclústeres: el **subclúster de almacenamiento** y el **subclúster de control**.

Los **N** nodos del clúster RAVD, tanto los del subclúster de almacenamiento, como los del de control se interconectan entre sí a través de una red privada en la cual se recomienda que se implemente LACP con al menos dos switches internos, donde estos trabajaran en modo "Link Agreggation".

Solo el nodo C maestro del clúster (y eventualmente su esclavo) es accedido desde la red de servicio a clientes finales.

La figura (28) a continuación, muestra el esquema de la estructura del clúster RAVD.

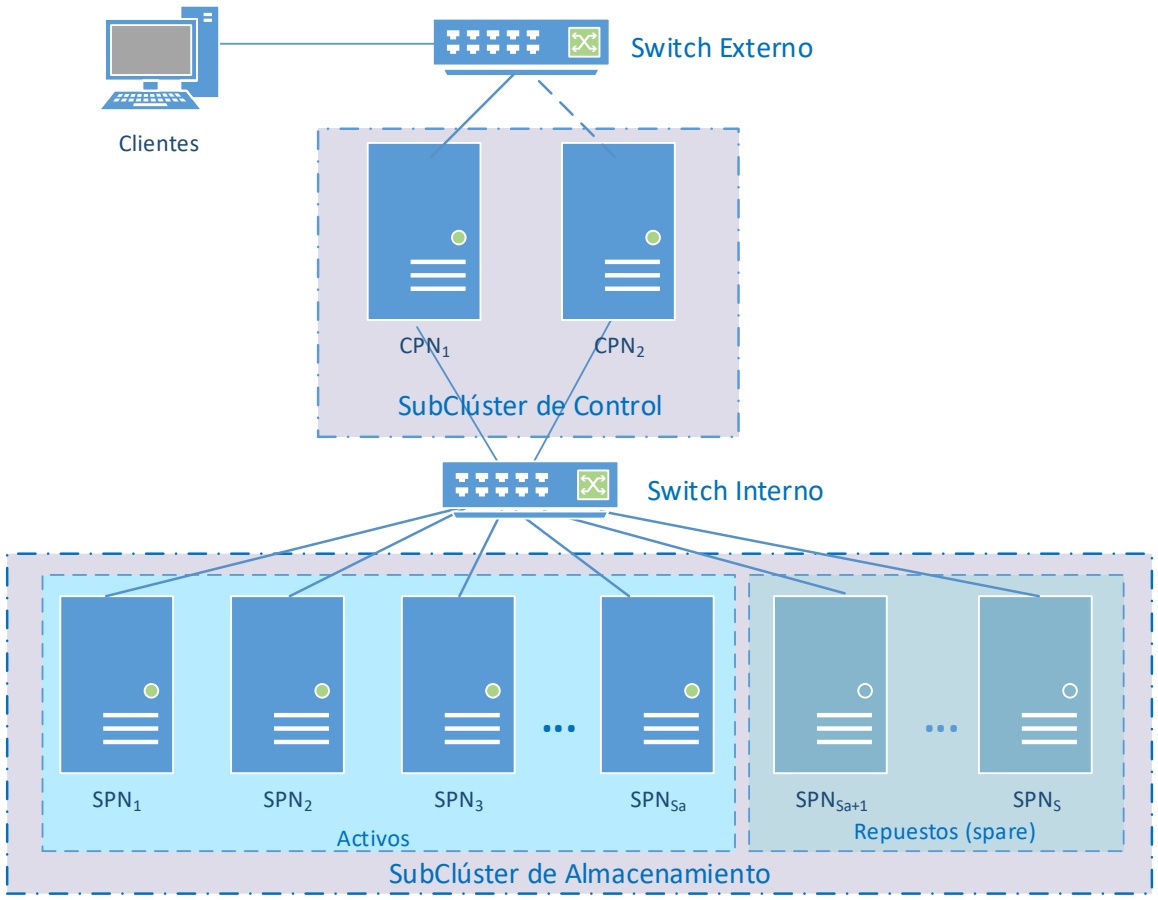


Figura (28) - Estructura del Clúster RAVD

La cantidad total **N** de nodos del clúster está definida por la siguiente expresión:

$$N = C + S = C + S_a + S_s$$

Donde:

- **C** es la cantidad de nodos de control
- **S** es la cantidad total de nodos de almacenamiento

- **S_a** es la cantidad de nodos de almacenamiento activos
- **S_s** es la cantidad de nodos de almacenamiento de repuesto (spare)

En la especificación de RAVD definida, C es constante e igual a dos. S_a debe ser mayor o igual a cuatro, y S_s es mayor o igual a cero.

A partir de lo anterior, la cantidad mínima de nodos del Clúster RAVD sin repuestos en línea es de **seis** nodos: dos nodos C redundantes en alta disponibilidad mediante modalidad maestro-esclavo y cuatro nodos S (requerimiento mínimo de RAID6) redundantes entre sí en alta disponibilidad con modalidad maestro-esclavo entre sus máquinas virtuales.

III.1.2.4. Arquitectura de servicios del RAVD

El modelo RAVD implementa un sistema completo de almacenamiento en línea mediante la integración de un conjunto de tecnologías que implementan servicios que interoperan sinérgicamente para proveer las diferentes funcionalidades requeridas para su operación.

Para que el subclúster de almacenamiento realice las funciones de:

a) servir un disco virtual al **MA-CVN** e
b) implementar alta disponibilidad entre sí,

cada instancia de servidor virtual (**MA-SVN** y **SL-SVN**) del **SPN** contiene las siguientes tecnologías de servicios configurados para trabajar de forma integrada en estrecha colaboración:

- **Dispositivo de bloque en red (*NBD: Network Block Device*), componente server.** Habilita acceso remoto al disco de la máquina virtual permitiendo servir un disco virtual al **MA-CVN**. **NBD** permite tratar bloques de datos remotos como si se tratasen de bloques de datos locales. Los servicios **NBD** de todos los **MA-SVN** convergen en el **MA-CVN** en forma

de un disco duro virtual cada uno, con lo cuales éste implementa el arreglo.

- **Réplica de bloques de dispositivos de forma remota (*DRBD: Distributed & Replicated Block Devices*)**. Replicación del disco de la máquina virtual **MA-SVN** en su correspondiente **SL-SVN**. El servicio de **DRBD** fue configurado para trabajar con el **protocolo C** (modalidad síncrona), el cual asegura confiabilidad en la replicación de los datos.
- **Alta disponibilidad entre dos ordenadores y monitoreo de servicios (*HEARTBEAT*)**. Implementa funcionalidad de clúster o alta disponibilidad entre servidores al habilitar mensajería y monitoreo de pertenencia entre sus clientes. A través de **HEARBEAT** los **SPN** se comunican entre sí y monitorean su disponibilidad. En caso de fallo de una **MA-SVN**, éste se encarga de promover su **SL-SVN** correspondiente para reemplazar el **MA-SVN** inmediatamente en virtud de que sus discos virtuales estaban sincronizados y replicados mediante **DRBD**.

Para que el subclúster de control realice las funciones de:

a) acceder remotamente a los discos virtuales de las **MA-SVN**
b) gestionar el arreglo de discos virtuales
c) implementar alta disponibilidad con su contraparte y
d) suministrar a los clientes el servicio de almacenamiento,

las instancias de servidor virtual (**MA-CVN** o **SL-CVN**) de los **CPN** contienen las siguientes tecnologías de servicios, configurados para trabajar de forma integrada en estrecha colaboración:

- **Dispositivo de bloque en red (*NBD: Network Block Device*), componente controlador**. Habilita acceso remoto al disco de la máquina virtual permitiendo acceder al disco virtual de las **MA-SVN**.
- **Administración de recursos y arreglos de dispositivos de bloques (*MDADM*).** Permite implementar y gestionar arreglos redundantes de

discos. Mediante **MDADM** se implementa el arreglo **RAID6** a partir de los discos remotos servidos por los **MA-SVN**.

- **Administración de sistemas de ficheros (*LVM, XFS*)**. Permite gestionar el espacio de almacenamiento del arreglo provisto por **MDADM**. Para la administración de la partición en unidades lógicas, se hace uso de las bondades de **LVM**, y para el sistema de archivos predeterminado se utiliza **XFS**.
- **Servicio de archivos (*NFS, FTP, etc.*).** Permite a los clientes finales del sistema de almacenamiento acceder al contenido del sistema. En caso de clientes de nube, se recomienda acceso vía **VPN**. Alternativamente se podrían implementar servicios de nube más sofisticados como **NextCloud**[39] u otro medio de compartición de archivos.
- **Alta disponibilidad entre dos ordenadores y monitoreo de servicios (*HEARTBEAT*)**. A través de **HEARBEAT** los **CPN** se comunican entre sí y monitorean su disponibilidad. En caso de fallo de la **MA-CVN**, éste se encarga de promover la **SL-CVN** a **MA-CVN**.

La figura (29) a continuación, muestra la arquitectura de servicios del Clúster RAVD, en la cual se puede apreciar la interacción de los diferentes elementos de éste, a través de los servicios requeridos para la realización de sus funciones.

[39] **NextCloud:** es un software compuesto por un conjunto de aplicaciones que combina la conveniencia y facilidad de uso al mismo nivel de facilidad como soluciones conocidas que son Dropbox y Google Drive, pero que no cumplen las necesidades comerciales de seguridad, privacidad y control sobre los datos almacenados.

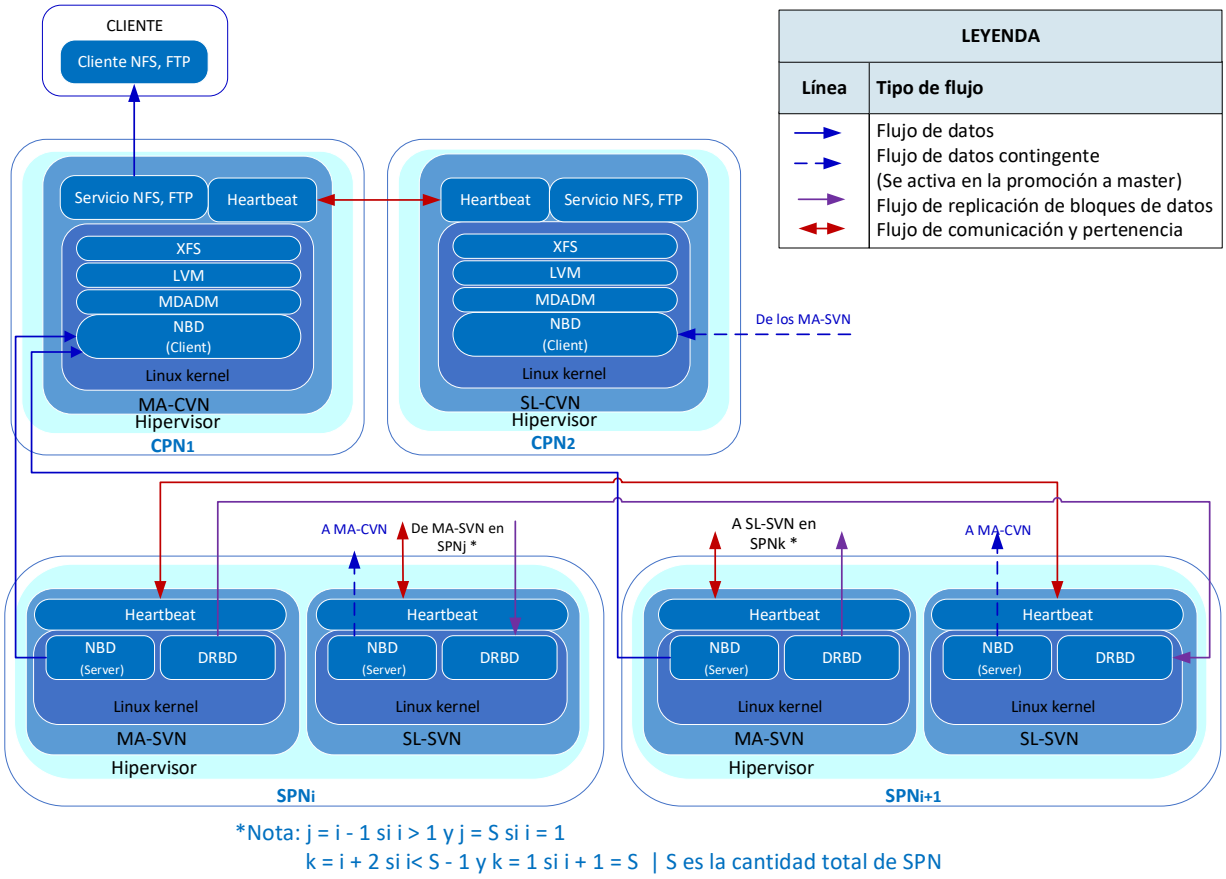


Figura (29) - Arquitectura de servicios del Clúster RAVD

La arquitectura de despliegue del sistema de respaldo en línea basado en RAVD se representada en el siguiente gráfico:

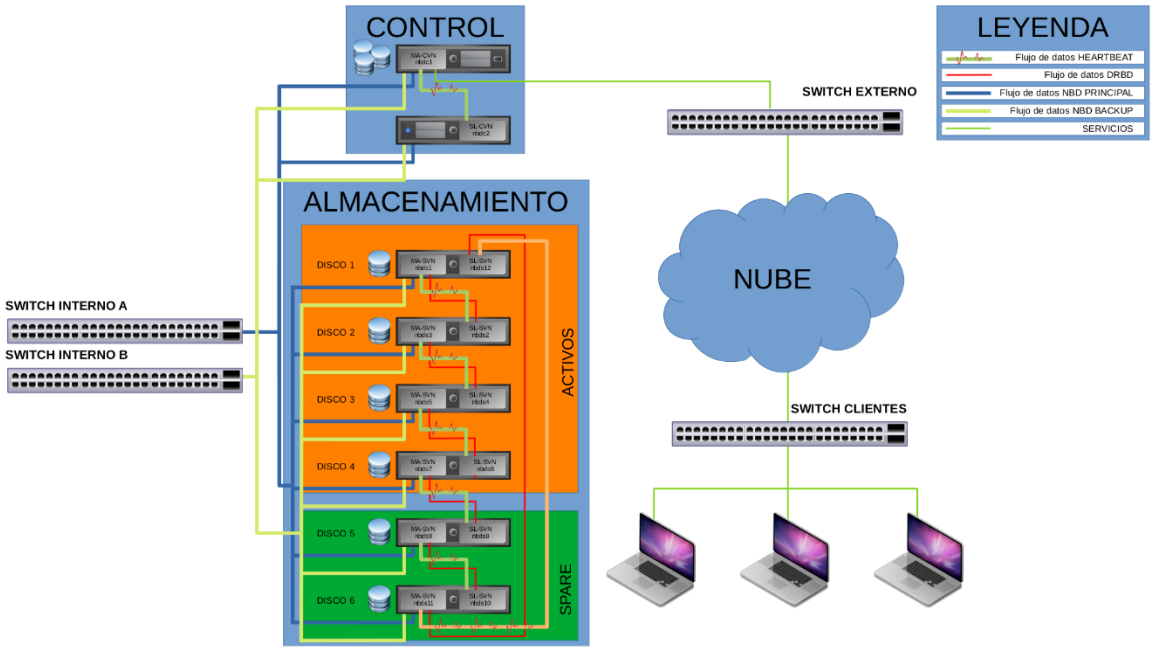


Figura (30) - Arquitectura de despliegue del Sistema de Respaldo en Línea basado en RAVD

III.1.3. Capacidad del Clúster RAVD

La capacidad total de almacenamiento que tendrá el clúster se calcula mediante la siguiente fórmula:

$$CTS = \left(\left(\frac{N - C}{Q}\right) - R - P\right) * SSD$$

Donde:

- **CTS** es la capacidad total de almacenamiento
- **N** es la cantidad de máquinas virtuales totales (SVN y CVN)
- **C** es la cantidad de CPN, con valor constante e igual a 2
- **Q** es la cantidad de réplicas de SVN, con valor constante e igual a 2
- **R** es la cantidad de discos de repuesto o spare
- **P** es el tamaño de la paridad según algoritmo RAID6, con valor constante igual a 2
- **SSD** es el tamaño de disco más pequeño de todos los SVN en el clúster

III.2. Requerimientos para implementar el sistema de almacenamiento

III.2.1. Requerimientos de Hardware

Los requerimientos mínimos de hardware necesario de cada nodo miembro del clúster en el modelo RAVD son los siguientes:

1. **Al menos seis (6) ordenadores con las siguientes características:**

- ✓ Disco duro mínimo 10 GB para el sistema y una partición a ser compartida.
- ✓ Procesadores que posean instrucciones tipo LAHF[40] y SAHF[41], ya que es un requerimiento por parte de los hipervisores a ser usado y requieren una de las siguientes características mínimas[42]:
  - Procesadores X86-64
    - Que posean 2 núcleos mínimos: Todos los procesadores Intel Xeon 5500/5600, 7100/7300, 7200/7400 y 7500 o sus sucesores.
    - Procesadores Intel Atom C2300 Series o superiores
  - Procesadores AMD Opteron.
- ✓ Memoria Interna mínima de 4 GB.
- ✓ De 2 a 4 puertos ethernet para conexiones de red.

---

[40] **LAHF:** (Load AH from Flags) copia de las banderas SF, ZF, AF, PF, y CF dentro del CPU a la bandera AH en los bits 7, 6, 4, 2 y 0 respectivamente. El contenido de bits faltantes (5,3 y 1) no están definidos. Las banderas permanecen intactas.

[41] **SAHF:** (Store AH into Flags) transfiere los bits 7, 6, 4, 2 y 0 de la bandera AH dentro de SF, ZF, AF, PF y CF respectivamente dentro del CPU.

[42] **Características mínimas para los procesadores:** para mayor referencia de las características mínimas necesarias se pueden consultar las siguientes url:

a) https://www.vmware.com/resources/compatibility/search.php.

b) https://www.linux-kvm.org/page/Processor_support

2. **Concentradores (Switch) de red con las siguientes características:**

- ✓ Puertos Ethernet con capacidad de Gigabit.
- ✓ Cantidad de puertos suficientes para conectar los ordenadores correspondientes, conforme rol:
  - Switch interno: conexión de los SPN.
  - Switch externo: conexión de los CPN.
- ✓ Interfaces físicas de conexión de dispositivos y de red de comunicaciones.

III.2.2. Requerimientos de Software

Los requerimientos mínimos de software necesario para la implementación del modelo RAVD, son los siguientes:

- ✓ Hipervisor (*ESXI o KVM*)
- ✓ Sistema operativo con licencia no comercial (*Debian 8.11 Jessie*)
- ✓ Aplicación para acceder un dispositivo de bloques distribuido (*NBD*)
- ✓ Aplicación para replicado de bloques distribuido (*DRBD*)
- ✓ Aplicación para gestión y monitoreo de arreglo de discos (*MDADM*)
- ✓ Administración de volúmenes y ficheros (*LVM, XFS*)
- ✓ Aplicación para habilitar alta disponibilidad mediante comunicación y monitoreo de pertenencia a clúster (*HEARTBEAT*)
- ✓ Aplicación para exponer una unidad lógica o volumen en red (*NFS, FTP, etc.*)

III.3. Selección de plataformas para implementación del modelo.

III.3.1. Hipervisor

El hipervisor seleccionado fue VMWare ESXI por las siguientes razones:

- ✓ Por ser el hipervisor utilizado por REKASA
- ✓ Por ser un hipervisor de tipo 1 (que se ejecuta a nivel de Kernel)

- ✓ Por ser líder del mercado debido a la estabilidad y seguridad que ofrece sobre otras opciones disponibles

Sin embargo, nuestro modelo funcionaría con cualquier otro tipo de hipervisor tipo 1, como por ejemplo KVM. (para mayor detalle de la selección del hipervisor ver Apéndice B).

III.3.2. Sistema Operativo

El sistema operativo usado fue Debian 8.11 (Jessie). Se eligió esta distribución de GNU/Linux, por cuanto es mantenida totalmente por una comunidad de voluntariado, y es respaldada por una organización, y no por una compañía, lo cual permite mayor libertad de uso y configuración (para mayor detalle de la selección del Sistema Operativo, ver Apéndice C).

III.4. Administración y mantenimiento del modelo RAVD

III.4.1. Sobre administración y mantenimiento del modelo RAVD

Se debe tomar en cuenta que pueden presentarse situaciones ocasionales donde se disparen alarmas de monitoreo, las cuales pueden conllevar a la planeación y calendarización de ventanas de mantenimiento para evitar fallos inminentes.

Si no existe un protocolo a seguir en el proceso de administración y mantenimiento del modelo RAVD, al igual que todo sistema de arreglo podría conllevar a la perdida de datos de forma irreversible.

Esta sección pretende prevenir o minimizar la posibilidad de perder los datos ante un fallo, o ante un procedimiento de mantenimiento.

III.4.2. Protocolo de contingencia

Se ha tratado de cubrir la mayor cantidad probable de fallos ante los cuales la empresa debe estar preparada para evitar la pérdida de datos que podrían darse

con el modelo RAVD. Los fallos que pueden llevar a la perdida de datos, y ante lo cual se debe trabajar con anticipación son los siguientes:

- **degradamiento del arreglo completo y pérdida total de los datos:** se sugiere tener un segundo respaldo mediante otro medio, ya sea usando una nube u otro proveedor.
- **fallo de energía:** la perdida de energización y el apagado de varias máquinas sería equivalente al fallo físico, lo cual conlleva a la degradación del arreglo y pérdida total de los datos, por lo cual cada ordenador físico debe tener energía redundante y de orígenes distintos. Debido a que en Nicaragua solo existe un único proveedor de energía comercial, es necesario obtener el segundo recurso energético por medio de baterías o generadores, para cuando el proveedor comercial falle o exista un corto de energía eléctrico en el lado comercial, los servidores continúen en línea por medio de los equipos alternos.
- **fallo de comunicación de red:** si los servidores dejan de comunicarse puede conllevar a la degradación del arreglo, por lo que la comunicación entre los servidores no debe depender de switches (ya que esto agrega puntos de fallo adicionales), lo que se recomienda es realizar una conexión directa entre los servidores, creando de esta forma una topología de anillo.

III.4.3. Protocolo de mantenimiento

Se plantean los fallos posibles que puedan darse para mitigar los daños que pueda sufrir el arreglo RAVD:

- **Para respaldos secundarios:**
  - El respaldo secundario puede tenerse en discos adicionales, externos o cintas magnéticas.
  - Se deben realizar respaldos periódicamente.
  - Debe revisarse la integridad de cada respaldo realizado.

- La empresa debe definir la realización de los respaldos completos o incrementales.
- El segundo medio de backup no necesita tener las mismas características del arreglo RAVD, pero si la misma capacidad para tener respaldo de los datos de este.

- **Pasos para reemplazar una VM:**
  - Se debe marcar la VM al estado averiado para luego ser retirada y excluida del arreglo presente.
  - Toda conexión y sesión establecida hacia la VM a ser retirada debe ser desconectada.
  - Es necesario que la VM a ser ingresada posea toda la paquetería y configuración de la VM retirada.
  - Una vez colocada en la red y visible por el Servidor Controlador, debe agregarse al arreglo como un disco en modo Spare.

- **Reemplazo de hardware físico:**
  - Se debe marcar la VM al estado averiado y ser excluida del arreglo.
  - La VM excluida debe desconectarse del Servidor Controlador y luego debe de ser apagada.
  - Se debe reemplazar la pieza dañada y se debe encender nuevamente y validar su estado de salud.
  - Una vez colocada en la red y visible por el Servidor Controlador, debe agregarse al arreglo como un disco en modo Spare.

- **Incremento de tamaño de arreglo:**
  - Se debe colocar la maquina visible en la red al Servidor Controlador.
  - Se debe desmontar el sistema de ficheros y la partición del arreglo montado.
  - Se debe agregar el nuevo disco virtual al arreglo.
  - Se debe modificar el sistema de archivos y reclamar el espacio sobrante.

- **Proceso para energía redundante:**
    - Debe haber dos tipos de baterías comerciales.
    - Los servidores deben poseer doble fuente de poder.
    - Cada fuente de poder debe poseer una fuente de energía distinta, aunque las baterías estén conectadas al mismo proveedor de energía.
    - Para servidores con una sola fuente de poder, se requiere de regletas PDU[43] con MBP[44] y ATS[45], componentes usados frecuentemente en Data Centers[46] para permitir la continuidad de energía en caso de fallos.

- **Continuidad en el enlace de red:**
    - Cada servidor debe poseer más de una tarjeta de red.
    - Cada tarjeta de red debe estar conectada a lo interno a un switch virtual interno.
    - Cada tarjeta de red debe estar conectada a lo externo a un switch físico externo.
    - Los switches físicos deben estar conectados en forma de anillo.

---

[43] **PDU (Power Distribution Unit):** en español Unidad de Energía Distribuida; es una barra de contactos altamente confiable, con múltiples tomacorrientes diseñada para suministrar energía regulada a equipos vitales de conexión en red, telecomunicaciones o servidores. A menudo se usa en combinación con un equipo de suministros de energía ininterrumpible (UPS)

[44] **MBP (Maintenance Bypass)**: en español Mantenimiento de desvío; son equipos que permiten la transferencia transparente de una carga eléctrica desde la energía del UPS a la energía de la red pública para una operación ininterrumpida de los equipos conectados cuando se realiza algún mantenimiento, ya sea reemplazo de las baterías o instalación de una nueva UPS

[45] **ATS (Automatic Transfer Switch):** en español Switch Automático de Transferencia; es un dispositivo que transfiere automáticamente una fuente de alimentación desde su fuente principal a una fuente de respaldo cuando detecta una falla o interrupción en la fuente principal

[46] **Para mayor información sobre continuidad energética en Data Centers**, puede revisar la siguiente URL: https://community.fs.com/blog/guide-to-power-distribution-unit.html

CAPITULO IV. IMPLEMENTACIÓN DEL SISTEMA DE ALMACENAMIENTO

IV.1. Alcance del prototipo

Para validar el diseño del sistema de respaldo de datos en línea basado en un clúster de almacenamiento en arreglo redundante de discos virtuales (RAVD), se implementó un prototipo funcional que incorpora los atributos requeridos por REKASA, limitado solamente en la cantidad de nodos (los mínimos requeridos por el modelo) y espacio de almacenamiento (10 GB por disco virtual).

Por tratarse de un prototipo, este se desarrolló en un ambiente controlado, el cual correspondió a un único equipo físico en el cual se implementaron todos los nodos "físicos" del clúster RAVD [2 CPN y 6 SPN (4 activos y 2 spare)] como grupos de máquinas virtuales, manteniendo la correspondencia conceptual y funcional con la arquitectura del RAVD desarrollada en el capítulo anterior.

En el caso de LACP, al desarrollarse el prototipo en un único equipo físico, no es posible implementarse esta configuración que requiere de switches reales, y tarjetas físicas para habilitar esta funcionalidad en la capa de red.

El prototipo se diseñó a fin de permitir validar todas las operaciones del sistema, así como su fiabilidad, disponibilidad y rendimiento.

A partir de la fórmula de la capacidad total de almacenamiento **CTS** descrita en el acápite **III.1.3. Capacidad del Clúster RAVD**, la capacidad del almacenamiento del prototipo es de **20 GB**, siendo que **N** es igual a catorce máquinas virtuales en total, **C** es igual a dos controladoras, **Q** es el factor de replicación de los **SVN** que es igual a dos, **R** que es la cantidad de discos spare el cual se definió en dos para el prototipo, y **P** es la paridad definida en **RAID6** y es igual a dos. La capacidad **SSD** de cada disco virtual se estableció nominalmente en **10GB** (aunque efectivamente queda en **9.99GB**), por tanto:

$$CTS = \left(\left(\frac{N-C}{Q}\right) - R - P\right) * SSD = \left(\left(\frac{14-2}{2}\right) - 2 - 2\right) * 9.99 = 19.98$$

IV.2. Arquitectura del prototipo

El prototipo implementa la arquitectura del clúster RAVD descrita en el capítulo III, garantizando la cantidad mínima de nodos requeridos:

a) **Subclúster de almacenamiento**: seis nodos, de los cuales cuatro se configuraron como activos y dos como spare.
b) **Subclúster de control:** dos nodos, uno como maestro y el otro como esclavo, conforme lo dispuesto por el modelo arquitectural del RAVD.

Para proveer los nodos del arreglo mediante un único equipo físico y un único hipervisor sobre éste, se implementaron catorce (14) máquinas virtuales, de las cuales:

a) **Dos (2)** rolan como **CVN** y sus correspondientes **MA-CVN** para el primero y **SL-CVN** para el segundo.
b) **Doce (12)** rolan como **SVN**, organizadas en conjuntos de dos máquinas virtuales que corresponden a un **SPN** cada uno (6 en total). En cada conjunto, una de ellas es **MA-SVN** y la otra, **SL-SVN**. De los seis (6) discos virtuales provistos por estos conjuntos **SPN** al **MA-CVN**, cuatro (4) se configuraron como activos en el arreglo y dos (2) como repuesto de inmediato en línea (spare).

El diseño del prototipo permite que trabajen todos los miembros orquestadamente, de forma idéntica a como lo harían en nodos físicos separados, gracias a las bondades que brindan las tecnologías que lo conforman, permitiendo la lectura y escritura de datos distribuidos, y al mismo tiempo brindando alta disponibilidad, y persistencia de datos.

El prototipo es estructuralmente equivalente al modelo RAVD sobre equipos físicos diferentes, con la salvedad de los puntos de fallo, pues si la máquina

anfitriona (física) del prototipo se daña, se pierden todas las máquinas virtuales, cuando, de haber utilizado ocho máquinas físicas, al perder una, sólo se perderían dos máquinas virtuales, si esta perteneciera al nodo de almacenamiento, y una sola máquina virtual si perteneciera al nodo controlador.

Los nodos del clúster fueron interconectados, mediante un switch interno virtual.

La figura (31) a continuación, muestra la estructura del prototipo del sistema de respaldo, el cual implementa el modelo del clúster RAVD en un único ordenador físico.

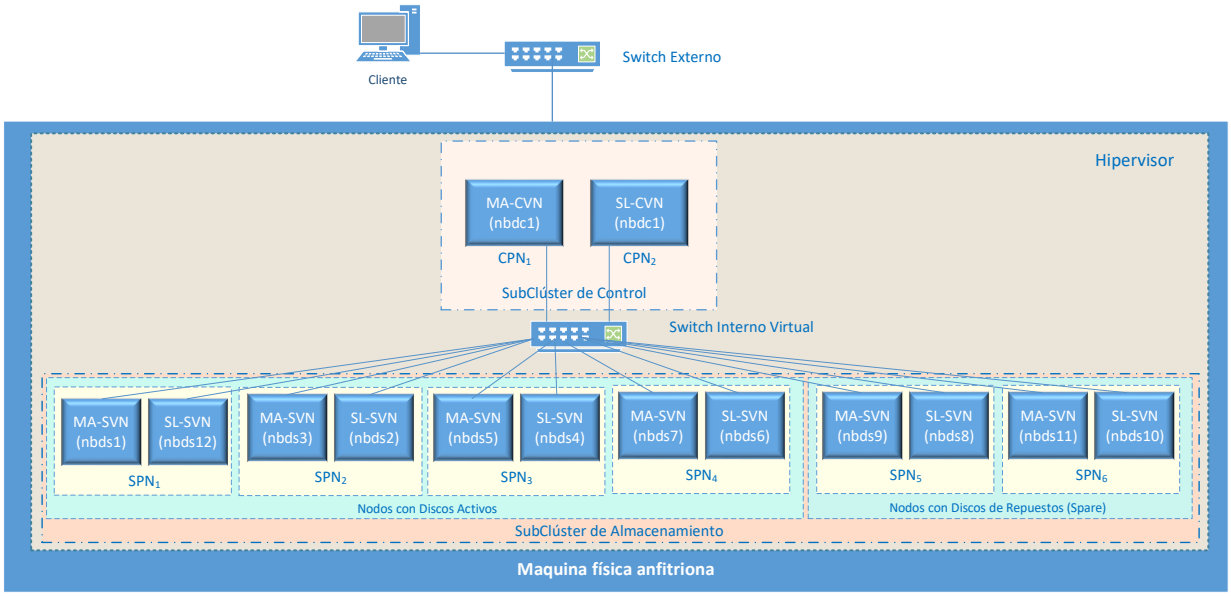


Figura (31) - Estructura del Prototipo del Sistema de Respaldo

La conexión entre las máquinas que conforman el arreglo de discos virtuales fue realizada directamente hacia un mismo switch virtual.

La figura (32) a continuación, muestra cómo se estructura el arreglo de discos RAVD a partir de la configuración del subclúster de Almacenamiento. Se ha sombreado con colores idénticos en las columnas “MÁQUINA VIRTUAL 1” y “MÁQUINA VIRTUAL 2”, las instancias de máquinas virtuales que están interrelacionados en modalidad maestro-esclavo.

Los valores de nbd**x**, donde **x** incrementa de uno en uno, empezando por el valor de cero, representan cómo quedan montados los bloques de datos que conforman el subclúster de almacenamiento implementado por MDADM en el subclúster de control como un arreglo de discos virtuales.

*: Discos en modalidad de SPARE

MÁQUINAS FÍSICAS	NÚMERO DE DISCO	FUNCIÓN		COMENTARIO	VERIFICACIÓN DEL ARREGLO
		MÁQUINA VIRTUAL 1 (MASTER)	MÁQUINA VIRTUAL 2 (SLAVE)		
SPN 1 [nbds1 - nbds12]	disco 1 [nbd0 (nbds1 - nbds2)]	MA-SVN d1 [nbds1 (nbd0)]	SL-SVN d6 [nbds12 (nbd5)] *	Arreglo funcional y estable	Number Major Minor RaidDevice State 0 43 0 0 active sync /dev/nbd0 1 43 16 1 active sync /dev/nbd1 2 43 32 2 active sync /dev/nbd2 3 43 48 3 active sync /dev/nbd3 4 43 64 - spare /dev/nbd4 5 43 80 - spare /dev/nbd5
SPN 2 [nbds3 - nbds2]	disco 2 [nbd1 (nbds3 - nbds4)]	MA-SVN d2 [nbds3 (nbd1)]	SL-SVN d1 [nbds2 (nbd0)]		
SPN 3 [nbds5 - nbds4]	disco 3 [nbd2 (nbds5 - nbds6)]	MA-SVN d3 [nbds5 (nbd2)]	SL-SVN d2 [nbds4 (nbd1)]		
SPN 4 [nbds7 - nbds6]	disco 4 [nbd3 (nbds7 - nbds8)]	MA-SVN d4 [nbds7 (nbd3)]	SL-SVN d3 [nbds6 (nbd2)]		
SPN 5 [nbds9 - nbds8]	disco 5 [nbd4 (nbds9 - nbds10)]	MA-SVN d5 [nbds9 (nbd4)] *	SL-SVN d4 [nbds8 (nbd3)]		
SPN 6 [nbds11 - nbds10]	disco 6 [nbd5 (nbds11 - nbds12)]	MA-SVN d6 [nbds11 (nbd5)] *	SL-SVN d5 [nbds10 (nbd4)] *		

Figura (32) - Estructura del Arreglo de Discos Virtuales RAVD sobre el subclúster de almacenamiento

IV.3. Instalación del hipervisor y máquina virtual

Según lo explicado anteriormente, el hipervisor seleccionado fue VMWare ESXI como propuesta para el modelo RAVD.

Debido a la imposibilidad de instalar VMWare ESXI por falta de hardware en los ordenadores con los que se contaron para probar el prototipo modelado, fue seleccionado VMWare Workstation para simular el modelo, además de existir portabilidad esto presentaría una aproximación simulada de un modelo al prototipo planteado:

1. **VMWare Workstation:** Hipervisor de tipo 2 que permite trabajar con equipos en desarrollo y estaciones de trabajo en ambiente controlado para pruebas y desarrollo, pero no para ambientes de producción.

La instalación de una máquina virtual bajo este software esta descrita en Apéndice D.

2. **VMWare ESXI:** Hipervisor de tipo 1 que permite trabajar con equipos dedicados para ambientes de producción, además de poseer un firmware muy pequeño, también está dedicado a la virtualización.

La instalación de una máquina virtual bajo este software esta descrita en Apéndice E.

IV.4. Instalación del Sistema Operativo

IV.4.1. Notas técnicas sobre la instalación del Sistema Operativo

Para implementar el prototipo funcional del modelo se instaló el sistema Operativo GNU/Linux Debian en 14 máquinas virtuales, con recursos mínimos para ejecutarlos todos en un solo anfitrión.

Cada máquina virtual fue configurada con recursos de hardware mínimos de 1024 MB de Memoria Interna, 1 CPU y disco duros de 10G, permitiendo al anfitrión soportar la cantidad de máquinas virtuales en ejecución a la vez (para este ejemplo).

Se denominó a los servidores controladores “nbdc” y a los servidores de almacenamiento “nbds”.

Se tomaron 2 máquinas virtuales para funcionar como controladores (una maestra y la otra esclava) y se les asignaron sus IP correspondientes:

- **nbdc1:** 172.16.2.10/24
- **nbdc2:** 172.16.2.11/24

Se emplearon las 12 máquinas virtuales restantes para los servicios (maestro/esclavo) y se les asignaron sus IP correspondientes:

- **nbds1:** 172.16.2.20/24
- **nbds2:** 172.16.2.21/24
- **nbds3:** 172.16.2.22/24
- **nbds4:** 172.16.2.23/24
- **nbds5:** 172.16.2.24/24
- **nbds6:** 172.16.2.25/24
- **nbds7:** 172.16.2.26/24
- **nbds8:** 172.16.2.27/24
- **nbds9:** 172.16.2.28/24
- **nbds10:** 172.16.2.29/24
- **nbds11:** 172.16.2.30/24
- **nbds12:** 172.16.2.31/24

Cuadro (001) - Tamaño del arreglo[47]

```
/dev/mapper/VGA-vwh     20G     11G   10G  51% /srv/disk
```

El valor obtenido es la capacidad del arreglo.

Cuadro (002) - Detalles del arreglo

```
/dev/md0:
        Version : 1.2
  Creation Time : Wed Aug 24 10:20:58 2022
     Raid Level : raid6
     Array Size : 23049216 (21.98 GiB 23.60 GB)
  Used Dev Size : 11524608 (10.99 GiB 11.80 GB)
   Raid Devices : 4
  Total Devices : 6
    Persistence : Superblock is persistent

    Update Time : Wed Aug 24 10:24:59 2022
          State : active
 Active Devices : 4
Working Devices : 6
 Failed Devices : 0
  Spare Devices : 2

         Layout : left-symmetric
     Chunk Size : 512K

           Name : nbdc1:0  (local to host nbdc1)
           UUID : d30c028c:d0968c3b:3a1449a6:f21adae5
         Events : 18

    Number   Major   Minor   RaidDevice State
       0      43        0        0      active sync   /dev/nbd0
       1      43       16        1      active sync   /dev/nbd1
       2      43       32        2      active sync   /dev/nbd2
       3      43       48        3      active sync   /dev/nbd3

       4      43       64        -      spare   /dev/nbd4
       5      43       80        -      spare   /dev/nbd5
```

[47] **Nota tipográfica 1:** en los cuadros de código de salida de consola, se resaltará en color amarillo la información más relevante a verificar de la salida de los comandos ejecutados.

IV.4.2. Realización de la instalación del Sistema Operativo

Teniendo la máquina virtual lista, y el disco para la instalación se instaló el sistema operativo, esta instalación fue idéntica para cada una de las máquinas, lo único que varió en cada una de ellas fue el hostname y la dirección IP.

Había dos formas de crear las máquinas virtuales:

1) Instalar el sistema operativo en cada máquina virtual de forma independiente y manual indicando en cada instalación el **IP** y el **hostname** que tendrá cada **máquina virtual**.
2) Instalar el sistema operativo en una sola **máquina virtual**, la cual posteriormente **será clonada** tantas veces como servidores virtuales se requieran, personalizando en cada una de ellas el **IP** y el **hostname** a utilizar.

La primera opción significaba invertir tiempo y esfuerzo, ya que se tendría que instalar el sistema operativo tantas veces como servidores virtuales se requieran de forma manual, repetitiva e idéntica.

Por lo explicado en el párrafo anterior, se decidió la segunda opción, porque significaba invertir menos tiempo y esfuerzo, se realizó la instalación de una sola máquina y se instalaron los paquetes base para no repetirlos en las demás máquinas virtuales, **exceptuando la instalación del DRBD y el HEARTBEAT**, los que solo deben de ser descargados antes de ser clonados. El clonado de máquinas virtuales es permitido gracias a las bondades que ofrece la virtualización.

IV.4.3. Implementación de ajustes o hacks[48] al Sistema Operativo.

Al haber instalado el Sistema Operativo, lo primero que hay que hacer son las configuraciones de afinamiento y personalización para tener comodidad al

[48] **Hack:** término usado en informática para referirse que se han realizado cambios en el código de cualquier software en pro a mejora de este.

momento de ir configurando, y realizando las pruebas requeridas como accesos, búsquedas, y bondades de las herramientas básicas como el editor de texto.

Primero, para permitir usar el historial en la consola mediante las teclas RePág y AvPág al trabajar con la línea de comandos, se le quitó el carácter "#" que indica comentario a las siguientes líneas en el archivo inputrc.

Cuadro (003) - Configuración del archivo inputrc[49]

Path: /etc/inputrc

```
...
# alternate mappings for "page up" and "page down" to search the history
"\e[5~": history-search-backward
"\e[6~": history-search-forward
...
```

Se configuró el editor vim para permitir la fácil lectura del contenido de un archivo, habilitando la detección de sintaxis, así como también la opción de indicarle que, al volver a abrir el editor, abra el archivo en el mismo lugar donde se dejó la edición al haberlo cerrado la última vez, para obtener lo mencionado anteriormente se descomentaron las siguientes líneas del archivo vimrc.

Cuadro (004) - Configuración del archivo vimrc

Path: /etc/vim/vimrc

```
...
syntax on
...
if has("autocmd")
  au BufReadPost * if line("'\"") > 1 && line("'\"") <= line("$") | exe
"normal! g'\"" | endif
endif
...
```

El siguiente archivo que se configuró, es el que define la información de red, el **IP, Máscara** y **Gateway** del servidor con el fin de ser identificado en la red, para ello se agregaron las siguientes líneas en el archivo interfaces.

[49] **Nota tipográfica 2:** en los cuadros de código de scripting o archivos de configuración como el presente, se emplearán puntos suspensivos para indicar que existe más código o texto en esa posición que no se presenta por no ser relevante para los efectos demostrativos deseados.

La interfaz eth0 da el servicio a internet, y la interfaz eth1 da servicio interno dentro del clúster.

Cuadro (005) - Configuración del archivo interfaces

Path: /etc/network/interfaces

```
...
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 172.16.2.10
netmask 255.255.255.0
```

La configuración para el resto de los servidores es idéntica, con la variante que los valores para "address" van cambiando para cada servidor según el IP que tendrá cada uno. (Ver configuración completa en Apéndice F)

Se configuró el archivo hosts de la siguiente forma en todas las máquinas virtuales, con el fin de que todos los ordenadores pudieran conocerse entre sí, para no modificar el archivo hosts, sería necesario un servidor de nombres interno:

Cuadro (006) - Configuración del archivo hosts

Path: /etc/hosts

```
...
172.16.2.10 nbdc1.proyecto.test nbdc1
172.16.2.11 nbdc2.proyecto.test nbdc2
172.16.2.20 nbds1.proyecto.test nbds1
172.16.2.21 nbds2.proyecto.test nbds2
172.16.2.22 nbds3.proyecto.test nbds3
172.16.2.23 nbds4.proyecto.test nbds4
172.16.2.24 nbds5.proyecto.test nbds5
172.16.2.25 nbds6.proyecto.test nbds6
172.16.2.26 nbds7.proyecto.test nbds7
172.16.2.27 nbds8.proyecto.test nbds8
172.16.2.28 nbds9.proyecto.test nbds9
172.16.2.29 nbds10.proyecto.test nbds10
172.16.2.30 nbds11.proyecto.test nbds11
172.16.2.31 nbds12.proyecto.test nbds12
```

Se autenticaron todos los miembros del clúster, esto se hace creando las llaves para cada servidor y publicarla en el resto, para esto creamos un script que se encarga de realizar la tarea y lo ejecutamos en cada servidor:

Cuadro (007) - Configuración llaves

Path: /root/copiarid.sh

```
#!/bin/bash
ssh-keygen -t rsa
ssh-add
for ip in $(grep nbd /etc/hosts| awk '{print $1}');do ssh-copy-id -i
~/.ssh/id_rsa.pub root@$ip;done
```

Para permitir el acceso root remoto es necesario modificar el archivo /etc/ssh/sshd_config:

Cuadro (008) - Permitiendo root externo para conexión ssh

/etc/ssh/sshd_config

```
Match Address 172.16.2.0/24
    PermitRootLogin yes
```

Debido a que las máquinas virtuales no poseen "floppy" y por defecto el sistema intenta cargarlo, se ejecutaron los siguientes comandos para omitirlo durante el arranque del sistema:

Cuadro (009) - Eliminar módulo floppy

```
# echo "blacklist floppy" > /etc/modprobe.d/floppy-blacklist.conf
# rmmod floppy
# update-initramfs –u
```

IV.5. Instalación de servicios

IV.5.1. Notas técnicas sobre la instalación de servicios

En esta sección se describen los detalles técnicos de los servicios y paquetes que fueron instalados.

IV.5.2. Realización de la instalación de los servicios

En Debian los paquetes fueron instalados con el comando aptitude o apt-get, por preferencia se usó el aplicativo aptitude.

Los paquetes instalados en los servidores controladores **nbdc1** y **nbdc2** fueron:

a) **vim:** editor de texto mejorado

b) **openssh-server:** programa para acceso remoto
c) **less:** paginador de archivo
d) **nbd-client:** aplicativo cliente para el servicio NBD
e) **git:** aplicativo para control de cambios y obtener o subir a un servidor git los archivos seleccionados
f) **heartbeat:** aplicación que administra la alta disponibilidad
g) **mdadm:** administrador de los arreglos RAID
h) **lvm2:** administra volúmenes lógicos
i) **nfs-common:** permite y da soporte a las herramientas cliente y servidor
j) **nfs-kernel-server:** permite y da soporte al kernel de las herramientas para compartir directorios en la red
k) **pv:** Programa de monitoreo de progreso de datos
l) **xfsprogs:** contiene un conjunto de comandos para el sistema de archivos **XFS**
m) **bash-completion:** programa de autocompletado
n) **vsftpd:** programa para transferencia de archivos

Cuadro (010) - Paquetes para nbdc1 y nbdc2

```
aptitude install vim less nbd-client git heartbeat mdadm lvm2 nfs-common
nfs-kernel-server openssh-server xfsprogs bash-completion pv vsftpd
```

Nota: no fue necesario instalar el servicio **DRBD** en las maquinas del subclúster Controlador: **nbdc1** y **nbdc2**, debido a que en estas máquinas no se necesita replicar bloques de discos (a diferencia de las máquinas **nbds1** a la **nbds12)**, puesto que su función solo es la de hacer público el arreglo del conjunto de discos recibidos**.**

La figura (33) a continuación, muestra cómo se estructura el subclúster Controlador donde será montado el arreglo de discos virtuales.

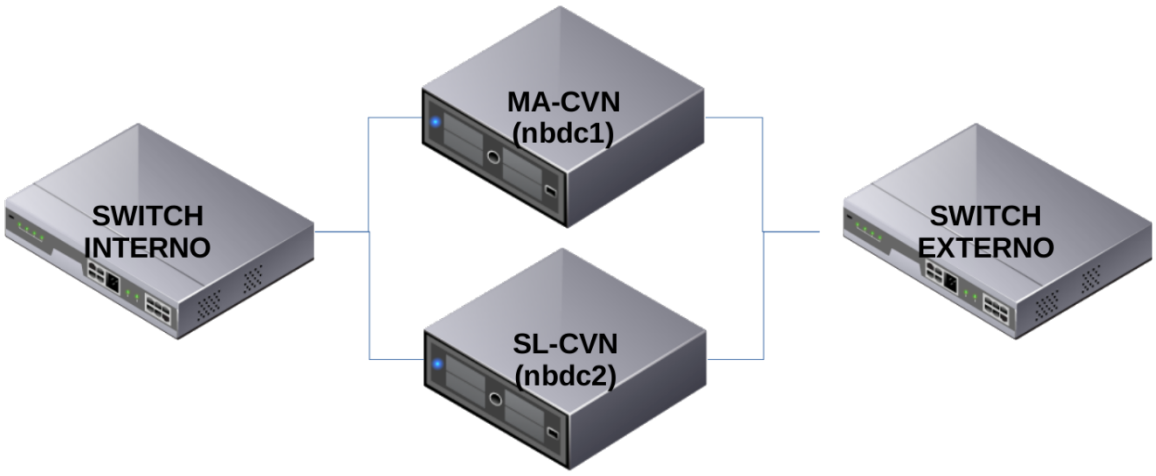


Figura (33) - Servidores NBDC

Los paquetes instalados en los servidores **nbds1** al **nbds12** fueron:

a) **vim:** editor de texto mejorado
b) **openssh-server:** programa para acceso remoto
c) **rsync:** programa para sincronización remota
d) **less:** paginador de archivo
e) **drbd-utils:** herramientas administrativas para sincronizar dispositivos de bloques
f) **heartbeat:** aplicación que administra la alta disponibilidad
g) **pv:** tubería de consola que mide datos que pasan por ella
h) **nbd-server:** aplicativo servidor para el servicio NBD
i) **bash-completion:** programa de autocompletado

Cuadro (011) - Paquetes para nbds1 al nbds12

```
aptitude install vim openssh-server less rsync drbd-utils heartbeat pv
nbd-server bash-completion
```

IV.6. Configuración de servicios

IV.6.1. Notas técnicas sobre las configuraciones

En esta sección se describen los detalles técnicos de las configuraciones realizadas en los servicios que conforman la arquitectura del clúster RAVD.

IV.6.2. Configuración de los recursos DRBD

Según lo explicado en el punto "IV.5.2. Realización de la instalación de los servicios", el servicio **DRBD** no necesitó ser configurado en **nbdc1** ni en **nbdc2**.

Se editó el archivo global "global_common.conf", el cual contiene la información de las opciones de configuración para el servicio **DRBD** indicando las acciones a ser tomadas en caso de ciertos eventos y situaciones por el momento contemplaremos solo una opción:

PROTOCOL: se usó el protocolo C (síncrono).

Para ver más detalles de las opciones del archivo de configuración, puede remitirse al Apéndice G

Cuadro (012) - Archivo de configuración global_common.conf en nbds1

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}

common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
```

```
      }

      disk {
…
      }

      net {
        protocol C;
      }
}
```

El siguiente archivo que se editó fue el que contiene los recursos “r0.res” a ser usados y creados:

- **DEVICE:** nombre del dispositivo a ser creado para acceder al driver.
- **DISK:** dispositivo de bloque donde se escribirán los datos.
- **META-DISK:** como drbd almacenara las piezas de información y como se almacena en el área dedicada.

Este archivo contiene también los servidores y la IP indicada mediante la opción **ADDRESS.**

Cuadro (013) - Archivo de configuración r0.res en nbds1

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds1 {
    address   172.16.2.20:7789;
  }
  on nbds2 {
    address   172.16.2.21:7789;
  }
}
```

Se configuró en el servidor nbds2 los mismos servicios:

Cuadro (014) - Archivo de configuración global_common.conf en nbds2

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}
```

```
common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
      }

      disk {
…
      }

      net {
        protocol C;
      }
}
```

Se configuró el archivo de los recursos:

Cuadro (015) - Archivo de configuración r0.res en nbds2

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds1 {
    address   172.16.2.20:7789;
  }
  on nbds2 {
    address   172.16.2.21:7789;
  }
}
```

El resto de las configuraciones en los demás servidores **nbds3** al **nbds12** son similares, lo cual puede constatarse en el Apéndice G.

Una vez creados los archivos de configuración se procedió a activar los recursos una única vez, creando de esta forma el dispositivo de bloque indicado en la configuración "/dev/drbd1", estos pasos se realizaron en los servidores **nbds1** al **nbds12**.

Cuadro (016) - Preparación de recurso r0 para drbd para nbds1 al nbds12

```
drbdadm create-md r0
drbdadm up r0
```

La sincronización inicial se realizó en los dispositivos impares (nbds1, nbds3, nbds5, nbds7, nbds9 y nbds11), sin embargo, esta configuración pudo haberse realizado en los pares (nbds2, nbds4, nbds6, nbds8, nbds10 y nbds12), siempre teniendo en cuenta que el par o impar escogido sincronice al otro que no se toma en cuenta en dicha configuración, ya que esta tarea debe hacerse solo en uno de los dos servidores.

Los pasos de la sincronización inicial están dados de la siguiente manera:

Cuadro (017) - Preparación de partición drbd para nbds1, nbds3, nbds5, nbds7, nbds9 y nbds11

```
echo "# Hacer primario el recurso para sincronizarlo con el secundario"
drbdadm primary --force r0

echo "# Devolver a secundario el recurso ya sincronizado"
drbdadm secondary r0

echo "# Reglas UDEV para disco sdb1 y drbd1"
cat << EOF >> /etc/udev/rules.d/sdb.rules
SUBSYSTEM=="block",       ENV{DEVNAME}=="/dev/sdb1",       OWNER="nbd",
GROUP="nbd", MODE="0660"
SUBSYSTEM=="block",      ENV{DEVNAME}=="/dev/drbd1",      OWNER="nbd",
GROUP="nbd", MODE="0660"
EOF

echo "# Recargar reglas udev"
udevadm control --reload-rules && udevadm trigger
```

IV.6.3. Configurando NBD

La configuración del servicio nbd en los servidores controladores se realizó modificando el archivo nbdc-client en los siguientes parámetros:

- **NBD_DEVICE[N]:** nombre de la entrada del dispositivo a ser usado
- **NBD_TYPE[N]:** tipo del dispositivo: s (swap), f (sistema de archivos), r (puro); se usó el tipo 'r'
- **NBD_HOST[N]:** servidor del cual los procesos provienen
- **NBD_NAME[N]:** nombre de la exportación, este valor debe de ser el mismo valor configurado en la configuración servidor

Cada parámetro funciona como un arreglo ya que pueden conectarse múltiples servidores, este valor puede tomar un valor **N,** representando este valor el servidor **nbds**, que se desea conectar al controlador.

Cuadro (018) - Configuración del archivo nbd-client en nbdc1

Path: /etc/nbd-client

```
...
NBD_DEVICE[0]=/dev/nbd0
...
NBD_TYPE[0]="r"
...
NBD_HOST[0]="172.16.2.50"
...
NBD_PORT[0]=
...
NBD_NAME[0]="disk1"
...
NBD_EXTRA[0]="-persist"
...
NBD_DEVICE[1]=/dev/nbd1
...
NBD_TYPE[1]="r"
...
NBD_HOST[1]="172.16.2.51"
...
NBD_PORT[1]=
...
NBD_NAME[1]="disk2"
...
NBD_EXTRA[1]="-persist"
...
NBD_DEVICE[2]=/dev/nbd2
...
NBD_TYPE[2]="r"
...
NBD_HOST[2]="172.16.2.52"
...
NBD_PORT[2]=
...
NBD_NAME[2]="disk3"
...
NBD_EXTRA[2]="-persist"
...
NBD_DEVICE[3]=/dev/nbd3
...
NBD_TYPE[3]="r"
...
NBD_HOST[3]="172.16.2.53"
...
NBD_PORT[3]=
...
NBD_NAME[3]="disk4"
...
```

```
NBD_EXTRA[3]="-persist"
...
NBD_DEVICE[4]=/dev/nbd4
...
NBD_TYPE[4]="r"
...
NBD_HOST[4]="172.16.2.54"
...
NBD_PORT[4]=
...
NBD_NAME[4]="disk5"
...
NBD_EXTRA[4]="-persist"
...
NBD_DEVICE[5]=/dev/nbd5
...
NBD_TYPE[5]="r"
...
NBD_HOST[5]="172.16.2.55"
...
NBD_PORT[5]=
...
NBD_NAME[5]="disk6"
...
NBD_EXTRA[5]="-persist"
```

Una vez configurado el servidor **nbdc1,** se configuró el servidor **nbdc2**. Se puede apreciar que la configuración es idéntica.

Cuadro (019) - Configuración del archivo nbd-client en nbdc2

Path: /etc/nbd-client

```
...
NBD_DEVICE[0]=/dev/nbd0
...
NBD_TYPE[0]="r"
...
NBD_HOST[0]="172.16.2.50"
...
NBD_PORT[0]=
...
NBD_NAME[0]="disk1"
...
NBD_EXTRA[0]="-persist"
...
NBD_DEVICE[1]=/dev/nbd1
...
NBD_TYPE[1]="r"
...
NBD_HOST[1]="172.16.2.51"
...
NBD_PORT[1]=
...
NBD_NAME[1]="disk2"
...
```

```
NBD_EXTRA[1]="-persist"
...
NBD_DEVICE[2]=/dev/nbd2
...
NBD_TYPE[2]="r"
...
NBD_HOST[2]="172.16.2.52"
...
NBD_PORT[2]=
...
NBD_NAME[2]="disk3"
...
NBD_EXTRA[2]="-persist"
...
NBD_DEVICE[3]=/dev/nbd3
...
NBD_TYPE[3]="r"
...
NBD_HOST[3]="172.16.2.53"
...
NBD_PORT[3]=
...
NBD_NAME[3]="disk4"
...
NBD_EXTRA[3]="-persist"
...
NBD_DEVICE[4]=/dev/nbd4
...
NBD_TYPE[4]="r"
...
NBD_HOST[4]="172.16.2.54"
...
NBD_PORT[4]=
...
NBD_NAME[4]="disk5"
...
NBD_EXTRA[4]="-persist"
...
NBD_DEVICE[5]=/dev/nbd5
...
NBD_TYPE[5]="r"
...
NBD_HOST[5]="172.16.2.55"
...
NBD_PORT[5]=
...
NBD_NAME[5]="disk6"
...
NBD_EXTRA[5]="-persist"
```

Una vez configurados los controladores **nbdc**, se configuraron lo servidores **nbds**. Los parámetros configurados del archivo de configuración son los siguientes:

- **USER:** indica el usuario bajo el cual el servicio ha de ser ejecutado

- **GROUP:** indica el grupo al que debe pertenecer el servicio que ha de ser ejecutado
- **INCLUDEDIR:** si se requieren incluir configuraciones extras se hacen desde esta carpeta agregando los archivos ahí
- **ALLOWLIST:** el valor es booleano (true o false) si es true permite que el cliente pueda pedir un listado de las exportaciones

Para exportar un recurso es requerido darle un nombre y encerrarlo entre corchetes ("**[ ]**"), este nombre es el configurado en el controlador bajo el parámetro **NBD_NAME[N]**, debe de ser compartido por los nodos **MA-SVN** miembro de un **SPNX** y **SL-SVN** miembro de un **SPNY**, ambos encargados de ofrecer el mismo recurso. Por ejemplo, teniendo **nbds1** y **nbds2**, estos compartirían el recurso **[disk1]**, cuando ocurra un fallo en el **MA-SVN**, los servicios serán levantados automáticamente por el **SL-SVN**, levantará el servicio bajo el mismo nombre **[disk1]** al tener la misma configuración, desde la perspectiva del **MA-CVN** el servicio permanece intacto.

El servicio se configuró editando los siguientes parámetros:

AUTHFILE: define una lista de servidores autorizados a usar el servicio.

EXPORTNAME: aquí se define el nombre del archivo a ser exportado. Archivo de bloques creado anteriormente con ruta /dev/drbd1.

Nota: al tener el primer archivo dd.img, este puede ser copiado a los demás servidores, con el fin de no tener que repetir el paso explicado anteriormente en cada uno de los servidores.

Cuadro (020) - Configuración del archivo config en nbds1

Path: /etc/nbd-server/config

```
[generic]
...
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true
...
```

```
      [disk1]
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Se creó el archivo de configuración "/etc/nbd-server/allow", el cual contiene los ordenadores autorizados a conectarse o establecer conexión por IP al servicio.

Cuadro (021) - Configuración del archivo allow en nbds1

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

Puede observarse que el servidor **nbds2** posee la misma configuración que **nbds1** donde el nombre de nuestro recurso es **"[disk1]"**, para permitir la continuidad del servicio al momento de una caída:

Cuadro (022) - Configuración del archivo config en nbds2

Path: /etc/nbd-server/config

```
[generic]
# If you want to run everything as root rather than the nbd user, you
# may either say "root" in the two following lines, or remove them
# altogether. Do not remove the [generic] section, however.
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true

# What follows are export definitions. You may create as much of them
as
# you want, but the section header has to be unique.
      [disk1]
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Se configuró igual que en **nbds1** los ordenadores autorizados a conectarse a **nbds2**:

Cuadro (023) - Configuración del archivo allow en nbds2

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

Habiendo configurado el recurso **"[disk1]"** en **nbds1** y **nbds2** se procedió a configurar **"[disk2]"** en **nbds3** y **nbds4**:

Cuadro (024) - Configuración del archivo config en nbds3

Path: /etc/nbd-server/config

```
[generic]
# If you want to run everything as root rather than the nbd user, you
# may either say "root" in the two following lines, or remove them
# altogether. Do not remove the [generic] section, however.
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true

# What follows are export definitions. You may create as much of them
as
# you want, but the section header has to be unique.
      [disk2]
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Nótese que el archivo allow es idéntico en todos los servidores **nbds**, aquí se configuró en el servidor **nbds3** tal como se configuraron en los servidores **nbds1** y **nbds2**:

Cuadro (025) - Configuración del archivo allow en nbds3

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

Se configuró el servidor **nbds4** igual que su par **nbds3**:

Cuadro (026) - Configuración del archivo config en nbds4

Path: /etc/nbd-server/config

```
[generic]
# If you want to run everything as root rather than the nbd user, you
# may either say "root" in the two following lines, or remove them
# altogether. Do not remove the [generic] section, however.
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true

# What follows are export definitions. You may create as much of them
as
# you want, but the section header has to be unique.
      [disk2]
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Se configuró el archivo allow igual que los anteriores:

Cuadro (027) - Configuración del archivo allow en nbds4

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

Nota: El resto de las configuraciones se incorporan en el Apéndice H.

IV.6.4. Configuración de HEARTBEAT

Se procedió a configurar el heartbeat iniciando con el archivo de la autenticación llamado **authkeys** que tiene la llave precompartida para la autenticación del nodo, esta debe tener permiso de lectura únicamente para el root conteniendo lo siguiente:

- **NUM:** es un simple índice de la llave, iniciando con 1. Usualmente solo existirá una sola llave en el archivo.
- **ALGORITHM:** esto indica el tipo de algoritmo a ser usado entre los cuales pueden ser MD5[50] o SHA1[51].
- **SECRET:** es la llave de autenticación usada.

Cuadro (028) - Configuración authkeys

```
auth <num>
<num> <algorithm> <secret>
```

[50] **MD5:** en criptografía, MD5 (abreviatura de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) es uno de los algoritmos de reducción criptográficos de 128 bits diseñados por el profesor Ronald Rivest del MIT (Massachusetts Institute of Technology, Instituto Tecnológico de Massachusetts). Fue desarrollado en 1991 como reemplazo del algoritmo MD4 después de que Hans Dobbertin descubriese su debilidad. Ver detalles en referencia (Rivest, 1992)

[51] **SHA:** (Secure Hash Algorithm, Algoritmo de Hash Seguro) es una familia de funciones hash de cifrado publicadas por el Instituto Nacional de Estándares y Tecnología (NIST). La primera versión del algoritmo fue creada en 1993 con el nombre de SHA, aunque en la actualidad se la conoce como SHA-0 para evitar confusiones con las versiones posteriores. La segunda versión del sistema, publicada con el nombre de SHA-1, fue publicada dos años más tarde. Ver detalles en referencia (Eastlake 3rd & Jones, 2001)

Este archivo puede ser generado ejecutando la siguiente línea de comandos:

Cuadro (029) - Generando Authkeys

```
( echo -ne "auth 1\n1 sha1 "; \
  dd if=/dev/urandom bs=512 count=1 | openssl sha1 ) \
  > /etc/ha.d/authkeys
chmod 0600 /etc/ha.d/authkeys
```

Cuadro (030) - Archivo de configuración authkeys en nbdc1

Path: /etc/heartbeat/authkeys

```
…
auth 1
1 sha1 (stdin)= 8cae6ac6bbd218ba814f07091ae8c8e0
```

Cuadro (031) - Archivo de configuración authkeys en nbdc2

Path: /etc/heartbeat/authkeys

```
…
auth 1
1 sha1 (stdin)= 8cae6ac6bbd218ba814f07091ae8c8e0
```

Cuadro (032) - Archivo de configuración authkeys en nbds1

Path: /etc/heartbeat/authkeys

```
…
auth 1
1 sha1 (stdin)= e9f38fc7336c80fdc0902beab347d20c92445036
```

Cuadro (033) - Archivo de configuración authkeys en nbds2

Path: /etc/heartbeat/authkeys

```
…
auth 1
1 sha1 (stdin)= e9f38fc7336c80fdc0902beab347d20c92445036
```

El archivo de configuración **ha.cf**, contiene los siguientes parámetros:

- **DEBUGFILE:** archivo donde se almacenará los mensajes de depuración
- **LOGFILE:** archivo donde se almacenará los mensajes de registro
- **LOGFACILITY:** instalación a ser usada para el registro
- **KEEPALIVE:** indica la periodicidad entre latidos
- **DEADTIME:** indica el tiempo a considerar para indicar que este caído
- **WARNTIME:** indica el tiempo antes de emitir la advertencia "finales de los latidos"
- **INITDEAD:** señala el primer tiempo muerto

- **UDPPORT:** indica el puerto UDP[52] a ser usado para la comunicación entre los miembros del nodo
- **UCAST:** configura el unicast[53]
- **AUTO_FAILBACK:** determina si un recurso al fallar devolverá los recursos al nodo primario
- **NODE:** indica al miembro del clúster

Cuadro (034) - Archivo de configuración ha.cf en nbdc1

Path: /etc/heartbeat/ha.cf

```
debugfile /var/log/ha.debug
logfile /var/log/ha.log
logfacility     local0
keepalive 1
deadtime 15
warntime 10
initdead 30
udpport 694
ucast eth0 172.16.2.11
auto_failback off
node nbdc1
node nbdc2
```

Cuadro (035) - Archivo de configuración ha.cf en nbdc2

Path: /etc/heartbeat/ha.cf

```
debugfile /var/log/ha.debug
logfile /var/log/ha.log
logfacility     local0
```

[52] **UDP:** (User Datagram Protocol) es un protocolo del nivel de transporte basado en el intercambio de datagramas (Encapsulado de capa 4 Modelo OSI). Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. Tampoco tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros; y tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción. Ver detalles en referencia (Postel J. , 1980)

[53] **Unicast:** la unidifusión o difusión única (en inglés: unicast) es el envío de información desde un único emisor a un único receptor. Se contrapone a multicast o multidifusión (envío a ciertos destinatarios específicos, más de uno), broadcast (radiado o difusión, donde los destinatarios son todas las estaciones en la red) y anycast (el destinatario es único, uno cualquiera no especificado). Ver detalles en referencia (Tanenbaum, Fork, 2009)

```
keepalive 1
deadtime 15
warntime 10
initdead 30
udpport 694
ucast eth0 172.16.2.10
auto_failback off
node nbdc1
node nbdc2
```

Cuadro (036) - Archivo de configuración ha.cf en nbds1

Path: /etc/heartbeat/ha.cf

```
debugfile /var/log/ha.debug
logfile /var/log/ha.log
logfacility     local0
keepalive 1
deadtime 15
warntime 10
initdead 30
udpport 694
ucast eth0 172.16.2.21
auto_failback off
node nbds1
node nbds2
```

Cuadro (037) - Archivo de configuración ha.cf en nbds2

Path: /etc/heartbeat/ha.cf

```
debugfile /var/log/ha.debug
logfile /var/log/ha.log
logfacility     local0
keepalive 1
deadtime 15
warntime 10
initdead 30
udpport 694
ucast eth0 172.16.2.20
auto_failback off
node nbds1
node nbds2
```

El próximo archivo de configuración es “haresources”, que especifica los recursos que se configuran siguiendo la siguiente sintaxis:

Cuadro (038) - Configuración de haresources

```
node-name resource1 resource2 ... resourceN
```

- **NODE-NAME:** indica el nombre del nodo.
- **RESOURCE:** indica el recurso o servicio a ser lanzado.

Cuadro (039) - Archivo de configuración haresources en nbdc1

Path: /etc/heartbeat/haresources

```
…
nbdc1 array_services 172.16.2.15
```

Cuadro (040) - Archivo de configuración haresources en nbdc2

Path: /etc/heartbeat/haresources

```
…
nbdc1 array_services 172.16.2.15
```

Cuadro (041) - Archivo de configuración haresources en nbds1

Path: /etc/heartbeat/haresources

```
…
nbds1 drbddisk::r0 172.16.2.50 nbd-server
```

Cuadro (042) - Archivo de configuración haresources en nbds2

Path: /etc/heartbeat/haresources

```
…
nbds1 drbddisk::r0 172.16.2.50 nbd-server
```

Nota: el resto de las configuraciones son similares, pueden ser consultadas en el Apéndice I.

IV.7. Creación del RAID6

Una vez configurados los servicios y teniendo los discos en los servidores controladores **nbdc1** y **nbdc2**, se creó el arreglo en modo RAID6.

Cuadro (043) - Configuración RAID6

```
mdadm --create /dev/md0 --level=6 --raid-devices=4 -spare-devices=2
/dev/nbd0 /dev/nbd1 /dev/nbd2 /dev/nbd3 /dev/nbd4 /dev/nbd5
```

Para guardar los cambios del arreglo creado anteriormente, se ejecutó el siguiente cambio:

Cuadro (044) - Guardando el arreglo en el archivo de configuración mdadm.conf

```
mdadm --examine --scan >> /etc/mdadm/mdadm.conf
```

Con esta operación se obtiene el arreglo redundante de discos virtuales.

El archivo de configuración fue copiado al servidor nbdc2 mediante el siguiente comando:

Cuadro (045) - Copiamos el archivo de configuración del arreglo

```
# scp /etc/mdadm/mdadm.conf nbdc2:/etc/mdadm/mdadm.conf
```

IV.8. Creación del volumen lógico con LVM

Antes de darle al arreglo formato, es necesario crear las particiones y esto se realiza con **LVM**:

Cuadro (046) - Configuración de LVM

```
pvcreate /dev/md0
vgcreate VGA /dev/md0
lvcreate -l 100%VG -n vwh VGA
```

IV.9. Formateo de la partición

El disco creado con **LVM** recibirá el formato **XFS** y se dio formato esta partición mediante el siguiente comando:

Cuadro (047) - Dándole formato a la partición

```
mkfs.xfs /dev/VGA/vwh
```

IV.10. Creación de partición para el montaje del arreglo

Cuadro (048) - Creando carpeta destino en servidores nbdc1 y nbdc2

```
mkdir /srv/disk/
```

IV.11. Configuración de servicio FTP

Se instaló el servicio **VSFTPD**, servicio que permitirá compartir los archivos. En los servidores nbdc1 y nbdc2 se realizaron los siguientes pasos.

Cuadro (049) - Ajustando archivo de configuración vsfptd.conf

Path: /etc/vsftpd.conf

```
...
allow_writeable_chroot=YES
anon_mkdir_write_enable=NO
anon_other_write_enable=NO
anon_upload_enable=NO
anon_world_readable_only=NO
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd.chroot_list
chroot_local_user=YES
data_connection_timeout=620
ftpd_banner="FTP VSD"
hide_ids=YES
idle_session_timeout=600
local_max_rate=0
ls_recurse_enable=YES
pasv_enable=YES
pasv_max_port=2099
pasv_min_port=2000
port_enable=YES
userlist_deny=YES
userlist_enable=YES
vsftpd_log_file=/var/log/vsftpd.log
write_enable=YES
xferlog_file=/var/log/vsftpd.log
xferlog_std_format=NO
```

Cuadro (050) - Creando archivo vsftpd.chroot_list

```
root@nbdc1:~# touch /etc/vsftpd.chroot_list
```

Cuadro (051) - Ajustando posición de arranque

```
root@nbdc1:~# awk -F: '{print $1}' /etc/passwd > /etc/vsftpd.user_list
```

Cuadro (052) - Ajustando posición de arranque

```
root@nbdc1:~# useradd -s /bin/bash -d /srv/disk -m ravd
root@nbdc1:~# passwd ravd
```

IV.12. Puesta en funcionamiento

Para iniciar la primera vez el clúster, se deben iniciar primeramente los servidores **nbds**, una vez en servicio todos los servidores **nbds**, se iniciaron los servidores **nbdc**. El arranque fue realizado en el siguiente orden:

1. **nbds1**
2. **nbds2**
3. **nbds3**
4. **nbds4**
5. **nbds5**
6. **nbds6**
7. **nbds7**
8. **nbds8**
9. **nbds9**
10. **nbds10**
11. **nbds11**
12. **nbds12**
13. **nbdc1**
14. **nbdc2**

Es necesario ajustar el orden de arranque de los servicios nfs y heartbeat en los servidores nbdc1 y nbdc2, para ello se realizó la modificación en los encabezados de arranque agregando "heartbeat" en las opciones "Required-Start y Required-Stop":

Cuadro (053) - Ajustando posición de arranque

Path: /etc/init.d/nfs-kernel-server

```
#!/bin/bash

### BEGIN INIT INFO
# Provides:          nfs-kernel-server
# Required-Start:    heartbeat $remote_fs nfs-common $portmap $time
# Required-Stop:     heartbeat $remote_fs nfs-common $portmap $time
# Should-Start:      $named
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Kernel NFS server support
# Description:       NFS is a popular protocol for file sharing across
#                    TCP/IP networks. This service provides NFS server
#                    functionality, which is configured via the
#                    /etc/exports file.
### END INIT INFO
...
```

Para aplicar los cambios se realizaron los siguientes comandos:

Cuadro (054) - Actualización de servicios para aplicar cambios de orden de arranque

```
# update-rc.d nfs-kernel-server remove
# update-rc.d heartbeat remove
# systemctl enable nfs-kernel-server
# systemctl enable heartbeat
```

Se han creado Scripts para el arranque de los servicios los cuales pueden encontrarse en el Apéndice J.

IV.13. Procedimientos de gestión y mantenimiento

IV.13.1. Notas técnicas sobre procedimientos de gestión y mantenimiento

Una vez puesto en marcha el **RAVD**, como en todo sistema informático se podrían presentar escenarios que pongan en riesgo el desempeño y funcionamiento del prototipo, por lo tanto, es necesario contar con mecanismos o procedimientos para solventar dichos escenarios.

Dentro de los procedimientos no se describieron los pasos para la reducción del arreglo, debido a que no es el objetivo de la empresa REKASA, la cual pretende tener un modelo que sea incremental.

Ya que el remover un miembro del arreglo será solo en caso de fallos para su reemplazo o mantenimientos, no se describe el procedimiento para la reducción del arreglo.

Por lo antes expuesto no se detalló a nivel técnico cómo se conlleva a hacer este procedimiento, ya que se propone siempre el aumento del espacio y no a la reducción de este.

Si aún fuese necesario realizar este procedimiento, favor de buscar referencias en internet donde encontrará información técnica y muy detallada para la realización de este procedimiento, y en algunos casos hasta encontrará herramientas de auxilio como Gparted que ya realizan este procedimiento al alcance de un clic.

Algunas de las acciones de mantenimiento o fallos inesperados que pueden presentarse en el modelo **RAVD**, son los siguientes:

IV.13.2. Planificación de mantenimientos

Para el buen funcionamiento y la prevención de fallos, es requerido que se realicen mantenimientos programados de forma periódica que permitan evaluar la funcionalidad, estabilidad y rendimiento del prototipo, esto incluye las siguientes tareas:

- ✓ Revisión de logs.
- ✓ Revisión de estados físicos de tarjetas de red.
- ✓ Revisión de condiciones físicas.
- ✓ Revisión de estado físico y respaldos eléctricos.
- ✓ Revisión de integridad de datos.
- ✓ Pruebas de velocidad de transferencia de datos.

IV.13.3. Reinicio, Apagado y Encendido del RAVD

Si por motivos de mantenimiento es requerido reiniciar todo el clúster, se ha desarrollado un script que realiza la tarea de inicio y detención de los servicios para el clúster.

Nota: puede verificar el script mencionado en el Apéndice J.

IV.13.4. Fallo de un miembro del RAID6

Independientemente las razones por las cuales falle uno o dos discos, el algoritmo RAID6 las soporta, y al momento de que ocurra esto se deben sustituir los discos averiados.

Para sustituir un miembro dañado se deben seguir el siguiente procedimiento:

1) Identificar el dispositivo de bloques averiado:

Cuadro (055) - Comando para ver estado del arreglo

```
cat /proc/mdstat
```

2) Modificar la bandera “--fail” al dispositivo de bloque para marcarlo como dañado:

Cuadro (056) - Marcando unidad averiada y removiéndola del arreglo

```
mdadm --manage /dev/md0 --fail /dev/nbd# --remove /dev/nbd#
```

3) Agregar el dispositivo de bloques a reemplazar:

Cuadro (057) - Agregando disco reparado o repuesto al servidor nbdc

```
/sbin/nbd-client 172.16.2.X -N diskY 10809 /dev/nbd#
```

4) Agregar el dispositivo de bloques a reemplazar al arreglo md0:

Cuadro (058) - Agregando disco reparado o repuesto

```
mdadm --manage /dev/md0 --add /dev/nbd#
```

IV.13.5. Agregar un nuevo miembro al RAID6

Cuando se desea agregar un miembro para hacer crecer el clúster, se deben de seguir los siguientes pasos:

1) Adicionar el nuevo dispositivo de bloque:

Cuadro (059) - Agregando nuevo dispositivo de bloques

```
mdadm --manage /dev/md0 --add /dev/nbd#
```

2) Indicar el nuevo tamaño de cantidad de dispositivos en el arreglo:

Cuadro (060) - Creciendo el arreglo con la cantidad de dispositivos

```
mdadm --grow --raid-devices=$(cat /proc/mdstat | head -3 | tail -1 |
awk -F[ '{print $2}'| awk -F\/ '{print $1}') /dev/md0
```

3) Desmontar el directorio antes de redimensionar el sistema de archivo:

Cuadro (061) - Desmontando directorio

```
umount /dev/VGA/vwh
```

4) Indicar a LVM que use el nuevo espacio libre:

Cuadro (062) - Reclamar el 100% del espacio en el dispositivo de bloques

```
lvextend -l +100%FREE /dev/VGA/vwh
```

5) Redimensionar el sistema de archivos **XFS**:

Cuadro (063) - Redimensionando el sistema de archivos

```
xfs_growfs /dev/VGA/vwh
```

6) Montar el dispositivo de bloques:

Cuadro (064) - Montando dispositivo de bloques

```
mount /dev/VGA/vwh /srv/disk
```

CAPITULO V. EVALUACIÓN DEL PROTOTIPO DEL SISTEMA DE ALMACENAMIENTO IMPLEMENTADO

V.1. Diseño de las pruebas de ensayo

Dentro de este capítulo se incorporan los casos de pruebas realizadas para verificar la idoneidad del modelo para suplir los requerimientos de REKASA, las cuales fueron elaboradas para verificar el funcionamiento del sistema de almacenamiento en línea, su escalabilidad, así como validar la estabilidad e integridad del modelo RAVD ante fallos y corroborar su rendimiento:

V.1.1. Alta disponibilidad e integridad de datos:

a. Extracción o detención de uno o más miembros del clúster en frío
b. Extracción o detención de uno o más miembros del clúster en caliente
c. Pérdida del controlador maestro, donde está montado el arreglo de discos virtuales
d. Agregar un nuevo miembro al clúster RAVD

V.1.2. Rendimiento

a. Comparación de tiempo de escritura/lectura en disco para el modelo **RAVD** con diferentes capacidades de Memoria Interna.

Ocasionalmente se presentan situaciones donde existen alarmas de monitoreo, las cuales conllevan a tomar acciones de mantenimiento previamente calendarizados para evitar fallos inminentes, por lo que RAVD igual que un Storage real es propenso y vulnerable ante fallo completo del sistema.

Esporádicamente podría producirse un fallo inesperado de hardware, el cual atentaría contra la integridad de los datos; dichos fallos quedan latentes aún luego

del mantenimiento preventivo a los equipos, por lo cual es requerido un monitoreo constante del arreglo.

V.2. Resultado de pruebas

V.2.1. Pruebas de alta disponibilidad e integridad de datos:

V.2.1.1. Pérdida de uno o más miembros del subclúster de almacenamiento en frío

Los casos previamente mencionados fueron simulados, realizándose bajo el siguiente orden:

- Se detuvo el arreglo
- Se detuvo uno de los miembros del clúster
- Se reinició el miembro y el arreglo

Después de realizar los procesos mencionados anteriormente se obtuvieron los siguientes resultados:

- Se mantuvo la integridad de los datos

Antes de iniciar la prueba se corroboró el estado del clúster, y el estado funcional de sus recursos iniciales.

Los equipos involucrados en dicha prueba fueron los siguientes: nbdc1, nbdc2, nbds1 y nbds2.

Se verificó el estado del arreglo con el siguiente comando.

Cuadro (065) - Revisión rápida del arreglo del clúster

```
root@nbdc1:~# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid6 nbd0[6] nbd4[4](S) nbd5[5](S) nbd3[3] nbd2[2] nbd1[1]
      20955136 blocks super 1.2 level 6, 512k chunk, algorithm 2 [4/4]
[UUUU]

unused devices: <none>
```

Además del comando anterior que solo muestra una vista general, existe otro comando que muestra una vista más detallada sobre el estado del arreglo y sus miembros.

Cuadro (066) - Revisión detallada del arreglo del clúster

```
root@nbdc1:~# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sat Aug 22 18:44:01 2020
     Raid Level : raid6
     Array Size : 20955136 (19.98 GiB 21.46 GB)
  Used Dev Size : 10477568 (9.99 GiB 10.73 GB)
   Raid Devices : 4
  Total Devices : 6
    Persistence : Superblock is persistent

    Update Time : Sun Aug 23 00:04:57 2020
          State : clean
 Active Devices : 4
Working Devices : 6
 Failed Devices : 0
  Spare Devices : 2

         Layout : left-symmetric
     Chunk Size : 512K

           Name : nbdc1:0  (local to host nbdc1)
           UUID : a2b04716:b8696ed6:e89cd8f4:22eb3034
         Events : 168

    Number   Major   Minor   RaidDevice State
       6      43        0        0      active sync   /dev/nbd0
       1      43       16        1      active sync   /dev/nbd1
       2      43       32        2      active sync   /dev/nbd2
       3      43       48        3      active sync   /dev/nbd3

       4      43       64        -      spare   /dev/nbd4
       5      43       80        -      spare   /dev/nbd5
```

Se verifico el servicio nbdc en el servidor nbdc1

Cuadro (067) - Estado conexiones servicio nbd en nbdc1

```
root@nbdc1:~# ps aux |grep nbd-client | grep -v grep
root        1057  0.0  0.0   4232   104 ?          Ss   00:04   0:00
/sbin/nbd-client 172.16.2.50 -N disk1 10809 /dev/nbd0
root        1063  0.0  0.0   4232   100 ?          Ss   00:04   0:00
/sbin/nbd-client 172.16.2.51 -N disk2 10809 /dev/nbd1
root        1069  0.0  0.0   4232   104 ?          Ss   00:04   0:00
/sbin/nbd-client 172.16.2.52 -N disk3 10809 /dev/nbd2
root        1075  0.0  0.0   4232   104 ?          Ss   00:04   0:00
/sbin/nbd-client 172.16.2.53 -N disk4 10809 /dev/nbd3
```

```
root        1081  0.0  0.0   4232   104 ?          Ss   00:04   0:00
/sbin/nbd-client 172.16.2.54 -N disk5 10809 /dev/nbd4
root        1087  0.0  0.0   4232   104 ?          Ss   00:04   0:00
/sbin/nbd-client 172.16.2.55 -N disk6 10809 /dev/nbd5
```

Fue necesario validar que miembro del grupo nbdc tenía el IP virtual asignado, para dicho fin se ejecutó el siguiente comando.

Cuadro (068) - Estado de la red nbdc1 tiene los recursos, ya que se ve que posee la IP virtual en eth1:0

```
root@nbdc1:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:f4:51:db brd ff:ff:ff:ff:ff:ff
    inet 172.16.3.128/24 brd 172.16.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fef4:51db/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:f4:51:e5 brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.10/24 brd 172.16.2.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet 172.16.2.15/24 brd 172.16.2.255 scope global secondary eth1:0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fef4:51e5/64 scope link
       valid_lft forever preferred_lft forever
```

Se verifico el estado del servicio nbdc en el servidor nbdc2, y se constató su estado pasivo.

Cuadro (069) - Estado conexiones servicio nbd en nbdc2

```
root@nbdc2:~# ps aux |grep nbd | grep -v grep
root@nbdc2:~#
```

También, se consultó el recurso de red en el servidor esclavo.

Cuadro (070) - Estado de la red nbdc2 no tiene los recursos, ya que se ve que no posee la IP virtual en eth1

```
root@nbdc2:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:ce:85:c5 brd ff:ff:ff:ff:ff:ff
    inet 172.16.3.137/24 brd 172.16.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fece:85c5/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:ce:85:cf brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.11/24 brd 172.16.2.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fece:85cf/64 scope link
       valid_lft forever preferred_lft forever
```

Teniendo el resultado de los servidores nbdc, se procedió a verificar el estado de los servidores nbds iniciando por el estado del servicio drbd.

Una vez que se verificó el correcto estado de los miembros del arreglo, y que los servicios estén ejecutándose correctamente en los servidores controladores nbdc, se procedió a verificar el estado en los servidores nbds.

Se pudo constatar el estado **UpToDate** en el servicio drbd en ambos servidores nbds1 y nbds2.

Cuadro (071) - Revisando estado UpToDate en servidor nbds1

```
root@nbds1:~# cat /proc/drbd
version: 8.4.3 (api:1/proto:86-101)
srcversion: 6681A7B41C129CD2CE0624F

 1: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
    ns:3892580 nr:0 dw:1380 dr:11554121 al:4 bm:251 lo:0 pe:0 ua:0 ap:0
ep:1 wo:f oos:0
```

Cuadro (072) - Revisando estado UpToDate en servidor nbds2

```
root@nbds2:~# cat /proc/drbd
version: 8.4.3 (api:1/proto:86-101)
srcversion: 6681A7B41C129CD2CE0624F

 1: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C r-----
    ns:0 nr:3892580 dw:3892580 dr:0 al:0 bm:251 lo:0 pe:0 ua:0 ap:0
ep:1 wo:f oos:0
```

El siguiente recurso por observar fueron las IP

Cuadro (073) - El recurso de red lo posee el servidor nbds1

```
root@nbds1:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast
state DOWN group default qlen 1000
    link/ether 00:0c:29:c8:c7:d0 brd ff:ff:ff:ff:ff:ff
    inet 172.16.3.129/24 brd 172.16.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fec8:c7d0/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:c8:c7:da brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.20/24 brd 172.16.2.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet 172.16.2.50/24 brd 172.16.2.255 scope global secondary eth1:0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fec8:c7da/64 scope link
       valid_lft forever preferred_lft forever
```

Se revisa los recursos de red en el servidor nbds2.

Cuadro (074) - Recurso de red en el servidor nbds2

```
root@nbds2:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:8c:2c:f5 brd ff:ff:ff:ff:ff:ff
    inet 172.16.3.130/24 brd 172.16.3.255 scope global eth0
```

```
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe8c:2cf5/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:8c:2c:ff brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.21/24 brd 172.16.2.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe8c:2cff/64 scope link
       valid_lft forever preferred_lft forever
```

La siguiente etapa fue realizar la simulación de la extracción de uno de los miembros del clúster, esto se logró mediante la detención del servicio drbd en el servidor nbds1.

Al realizar la revisión, luego la detención y la revisión nuevamente del servicio se observó la transición y cambio del servicio y estado de los recursos:

Cuadro (075) - Se reinicia el servicio

```
root@nbds1:~# cat /proc/drbd
version: 8.4.3 (api:1/proto:86-101)
srcversion: 6681A7B41C129CD2CE0624F

 1: cs:Connected ro:Primary/Secondary ds:UpToDate/UpToDate C r-----
    ns:3892592 nr:8 dw:1400 dr:11555050 al:4 bm:251 lo:0 pe:0 ua:0 ap:0
ep:1 wo:f oos:0
root@nbds1:~# systemctl stop heartbeat
root@nbds1:~# cat /proc/drbd
version: 8.4.3 (api:1/proto:86-101)
srcversion: 6681A7B41C129CD2CE0624F

 1: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C r-----
    ns:3892596 nr:12 dw:1408 dr:11555050 al:4 bm:251 lo:0 pe:0 ua:0
ap:0 ep:1 wo:f oos:0
```

Luego de detener el servicio, se validó que el recurso ha sido transmitido hacia nbds2, se confirmó que nbds1 ya no posee eth1:0 y que ahora está en nbds2

Cuadro (076) - Revisión de recursos de red en nbds1

```
root@nbds1:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
```

```
    link/ether 00:0c:29:c8:c7:d0 brd ff:ff:ff:ff:ff:ff
    inet 172.16.3.129/24 brd 172.16.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fec8:c7d0/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:c8:c7:da brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.20/24 brd 172.16.2.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fec8:c7da/64 scope link
       valid_lft forever preferred_lft forever
```

Cuadro (077) - Descripción de recursos de red en nbds2

```
root@nbds2:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:8c:2c:f5 brd ff:ff:ff:ff:ff:ff
    inet 172.16.3.130/24 brd 172.16.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe8c:2cf5/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:8c:2c:ff brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.21/24 brd 172.16.2.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet 172.16.2.50/24 brd 172.16.2.255 scope global secondary eth1:0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe8c:2cff/64 scope link
       valid_lft forever preferred_lft forever
```

Se verifico el estado del arreglo en nbdc1 y se observó que el servidor no detecto el fallo

Cuadro (078) - Revisión rápida del estado del arreglo

```
root@nbdc1:~# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid6 nbd0[0] nbd4[4](S) nbd5[5](S) nbd3[3] nbd2[2] nbd1[1]
      20955136 blocks super 1.2 level 6, 512k chunk, algorithm 2 [4/4]
[UUUU]

unused devices: <none>
```

Cuadro (079) - Revisión detallada del arreglo

```
root@nbdc1:~# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sat Aug 22 18:44:01 2020
     Raid Level : raid6
     Array Size : 20955136 (19.98 GiB 21.46 GB)
  Used Dev Size : 10477568 (9.99 GiB 10.73 GB)
   Raid Devices : 4
  Total Devices : 6
    Persistence : Superblock is persistent

    Update Time : Sun Aug 23 00:04:57 2020
          State : clean
 Active Devices : 4
Working Devices : 6
 Failed Devices : 0
  Spare Devices : 2

         Layout : left-symmetric
     Chunk Size : 512K

           Name : nbdc1:0  (local to host nbdc1)
           UUID : a2b04716:b8696ed6:e89cd8f4:22eb3034
         Events : 168

    Number   Major   Minor   RaidDevice State
       6      43        0        0      active sync   /dev/nbd0
       1      43       16        1      active sync   /dev/nbd1
       2      43       32        2      active sync   /dev/nbd2
       3      43       48        3      active sync   /dev/nbd3

       4      43       64        -      spare   /dev/nbd4
       5      43       80        -      spare   /dev/nbd5
```

Con la prueba realizada y resultado obtenido, se certificó la posibilidad de realizar cambios de discos en ventanas previamente calendarizadas sin afectación en la integridad de los datos que era el propósito de esta prueba.

V.2.1.2.Pérdida de uno o más miembros del subclúster de almacenamiento en caliente

Las pruebas para simular fallos inesperados fueron realizadas en el siguiente orden:

- Se inició el proceso de escritura en el arreglo
- Se detuvo uno de los miembros del clúster
- Se reinició el miembro y el arreglo

Después de realizar los procesos mencionados anteriormente, se obtuvieron los siguientes resultados:

- Se detectó el fallo producido
- Se mantuvo la integridad de los datos

Con la prueba realizada y resultado obtenido, se confirmó la posibilidad de que puedan presentarse fallos inesperados en el arreglo, sin que estos afecten la integridad de los datos, que era el propósito de esta prueba.

Del par nbds1-nbds2 se simula caída del servidor nbds1, primera prueba, clúster en estado activo de escritura en el arreglo.

Se restablece el estado del clúster dejando nbds1 como perteneciente del recurso.

Cuadro (080) - Escritura de datos en el arreglo

```
root@nbdc1:~# dd if=/dev/zero | pv -s 7848591360 | dd of=/dev/drbd1
count=7664640 bs=1024
dd: warning: partial read (512 bytes); suggest iflag=fullblock       ]
18% ETA 0:02:10
```

Durante el copiado se confirma que los recursos estén en nbds1

Cuadro (081) - Revisión de recursos de red en nbds1

```
root@nbds1:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:c8:c7:d0 brd ff:ff:ff:ff:ff:ff
    inet 172.16.3.129/24 brd 172.16.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fec8:c7d0/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:c8:c7:da brd ff:ff:ff:ff:ff:ff
```

```
    inet 172.16.2.20/24 brd 172.16.2.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet 172.16.2.50/24 brd 172.16.2.255 scope global secondary eth1:0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fec8:c7da/64 scope link
       valid_lft forever preferred_lft forever
```

Se hace la simulación de caída de datos durante la escritura de datos.

Cuadro (082) - Reinicio de servicio, simulación de caída del equipo

```
root@nbds1:~# systemctl restart heartbeat
```

Se revisa el estado de los recursos estando en nbds2.

Cuadro (083) - Revisión de recursos de red en nbds2

```
root@nbds2:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:8c:2c:f5 brd ff:ff:ff:ff:ff:ff
    inet 172.16.3.130/24 brd 172.16.3.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe8c:2cf5/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP group default qlen 1000
    link/ether 00:0c:29:8c:2c:ff brd ff:ff:ff:ff:ff:ff
    inet 172.16.2.21/24 brd 172.16.2.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet 172.16.2.50/24 brd 172.16.2.255 scope global secondary eth1:0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe8c:2cff/64 scope link
       valid_lft forever preferred_lft forever
```

Se revisa el estado del clúster durante el copiado, luego de haber simulado el fallo, se verifico que el clúster detecto el fallo.

Durante las pruebas se verificó que algunas veces al provocar un fallo en nbds1, y por ende ser promovido el esclavo (nbds2) a maestro, durante un proceso activo de escritura, el arreglo de disco en RAID6 no admitía a nbds2, provocando que el bloque de datos nbd0 quedara en estado "**faulty**", y el bloque de datos ndb4 (spare) fuera puesto en funcionamiento para ser reconstruido mediante la paridad dual de

RAID6, dejando como parte del arreglo activo a nbds9 quien esta como maestro del bloque de datos ndb4.

Lo explicado anteriormente solo se presenta en escenarios donde un miembro activo del arreglo falla mientras se está dando un proceso de copia, al contrario, si al momento del fallo no se está realizando un proceso de copia, el esclavo que está siendo promovido a maestro es incorporado como miembro activo del arreglo sin ningún inconveniente, dejando a nbd0 (o el bloque de dato nbd**x** que este conformado por la máquina que presento el falló) siempre en estado "**active sync**" y a nbd4 (o cualquier primer nbd**x** en estado spare) su mismo estado "**spare**".

Cuadro (084) - Revisión detallada del arreglo

```
/dev/md0:
        Version : 1.2
  Creation Time : Sat Aug 22 18:44:01 2020
     Raid Level : raid6
     Array Size : 20955136 (19.98 GiB 21.46 GB)
  Used Dev Size : 10477568 (9.99 GiB 10.73 GB)
   Raid Devices : 4
  Total Devices : 6
    Persistence : Superblock is persistent

    Update Time : Sun Aug 23 02:40:23 2020
          State : clean, degraded, recovering
 Active Devices : 3
Working Devices : 5
 Failed Devices : 1
  Spare Devices : 2

         Layout : left-symmetric
     Chunk Size : 512K

 Rebuild Status : 74% complete

           Name : nbdc1:0  (local to host nbdc1)
           UUID : a2b04716:b8696ed6:e89cd8f4:22eb3034
         Events : 287

    Number   Major   Minor   RaidDevice State
       4      43       64        0      spare rebuilding   /dev/nbd4
       1      43       16        1      active sync        /dev/nbd1
       2      43       32        2      active sync        /dev/nbd2
       3      43       48        3      active sync        /dev/nbd3

       5      43       80        -      spare   /dev/nbd5
       6      43        0        -      faulty  /dev/nbd0
```

Cuadro (085) - Revisión rápida del arreglo

```
root@nbdc1:~# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid6 nbd0[6](F) nbd4[4] nbd5[5](S) nbd3[3] nbd2[2] nbd1[1]
      20955136 blocks super 1.2 level 6, 512k chunk, algorithm 2 [4/4]
[UUUU]

unused devices: <none>
```

Proceso de agregado del disco reparado

Cuadro (086) - Quitar disco dañado y agregar reparado

```
root@nbdc1:~# mdadm --manage /dev/md0 --fail /dev/nbd0 --remove
/dev/nbd0
root@nbdc1:~# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Sat Aug 22 18:44:01 2020
     Raid Level : raid6
     Array Size : 20955136 (19.98 GiB 21.46 GB)
  Used Dev Size : 10477568 (9.99 GiB 10.73 GB)
   Raid Devices : 4
  Total Devices : 5
    Persistence : Superblock is persistent

    Update Time : Sun Aug 23 03:00:37 2020
          State : clean
 Active Devices : 4
Working Devices : 5
 Failed Devices : 0
  Spare Devices : 1

         Layout : left-symmetric
     Chunk Size : 512K

           Name : nbdc1:0  (local to host nbdc1)
           UUID : a2b04716:b8696ed6:e89cd8f4:22eb3034
         Events : 298

    Number   Major   Minor   RaidDevice State
       4      43       64        0      active sync   /dev/nbd4
       1      43       16        1      active sync   /dev/nbd1
       2      43       32        2      active sync   /dev/nbd2
       3      43       48        3      active sync   /dev/nbd3

       5      43       80        -      spare   /dev/nbd5
root@nbdc1:~# mdadm --manage /dev/md0 --add /dev/nbd0
root@nbdc1:~# mdadm --detail /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Tue Sep 15 18:32:41 2020
     Raid Level : raid6
     Array Size : 20955136 (19.98 GiB 21.46 GB)
  Used Dev Size : 10477568 (9.99 GiB 10.73 GB)
```

```
   Raid Devices : 4
  Total Devices : 6
    Persistence : Superblock is persistent

    Update Time : Tue Sep 15 20:42:59 2020
          State : clean
 Active Devices : 4
Working Devices : 6
 Failed Devices : 0
  Spare Devices : 2

         Layout : left-symmetric
     Chunk Size : 512K

           Name : nbdc1:0  (local to host nbdc1)
           UUID : a543df4f:da38ebc2:2292ae78:0f16976d
         Events : 67

    Number   Major   Minor   RaidDevice State
       5      43       80        0      active sync   /dev/nbd4
       1      43       16        1      active sync   /dev/nbd1
       2      43       32        2      active sync   /dev/nbd2
       3      43       48        3      active sync   /dev/nbd3

       4      43       64        -      spare   /dev/nbd5
       6      43        0        -      spare   /dev/nbd0
```

V.2.1.3.Pérdida del controlador maestro del subclúster de control donde está montado el arreglo de discos virtuales

Esta prueba tuvo como objetivo verificar lo que sucede cuando se está realizando un proceso de copia o descarga desde un cliente final, y en el modelo RAVD el Controlador Maestro (donde está montado el arreglo de discos) falla.

Las pruebas para simular fallos inesperados en el Controlador Maestro fueron realizadas en el siguiente orden:

- Se inició el proceso de subida de un archivo mediante FTP al arreglo desde un cliente final
- Se detuvo el controlador maestro nbd activo.

Después de realizar los procesos mencionados anteriormente, se obtuvieron los siguientes resultados:

- Se detectó el fallo producido, y por medio de Heartbeat, el controlador esclavo fue promovido a maestro
- Al ser promovido el esclavo a maestro, todos los recursos pasaron a estar disponibles nuevamente, pero bajo la IP y Mac Address del nuevo controlador activo como maestro
- El proceso de subida del cliente final fue interrumpido por el fallo, por lo cual no se logró finalizar la tarea

Con la prueba realizada y resultados obtenidos, se validó la posibilidad de que durante un proceso operativo pueda fallar el controlador maestro, provocando la detención de la descarga o subida de datos, los cuales pueden ser reanudados nuevamente desde donde habían quedado al momento del fallo, gracias a que el controlador esclavo es promovido a maestro, suplantando la tarea del equipo que falló, y poniendo en disposición de esta forma todos los recursos nuevamente, sin que los datos se vean afectados, que era el propósito de esta prueba.

Se revisa los recursos y estado de servidores nbdc1 y nbdc2:

Cuadro (087) - Revisión rápida del estado de recursos de red en servidores nbdc1 y nbdc2

```
root@nbdc1:~# df -h
S.ficheros     Tamaño Usados  Disp Uso% Montado en
/dev/sda1        7.4G   1.3G  5.8G  18% /
udev              10M      0   10M   0% /dev
tmpfs             49M   4.5M   44M  10% /run
tmpfs            122M      0  122M   0% /dev/shm
tmpfs            5.0M      0  5.0M   0% /run/lock
tmpfs            122M      0  122M   0% /sys/fs/cgroup
/dev/mapper/VGA-vwh    22G   452M   22G   3% /srv/disk
root@nbdc2:~# df -h
S.ficheros     Tamaño Usados  Disp Uso% Montado en
/dev/sda1        7.4G   1.3G  5.8G  18% /
udev              10M      0   10M   0% /dev
tmpfs             49M   4.4M   45M  10% /run
tmpfs            122M      0  122M   0% /dev/shm
tmpfs            5.0M      0  5.0M   0% /run/lock
tmpfs            122M      0  122M   0% /sys/fs/cgroup
```

Para la prueba el primer paso fue establecer una conexión hacia el servidor nbdc1 con la IP 172.16.3.10

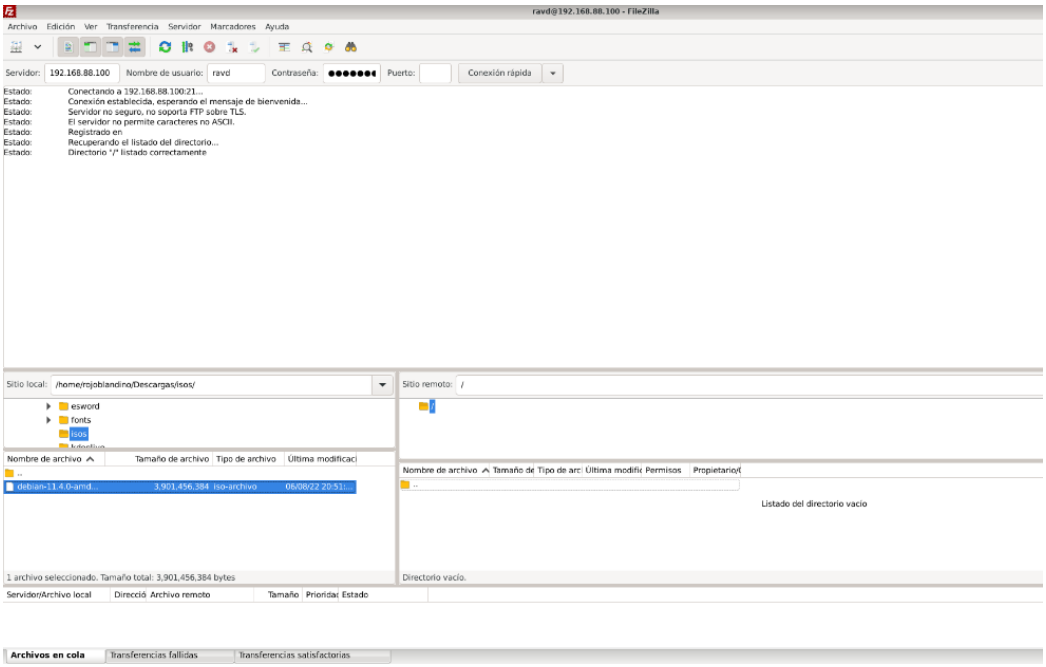


Figura (34) - Inicio de sesión a equipo Controlador Master NBD

Se inicia el proceso de prueba de transferencia de datos:

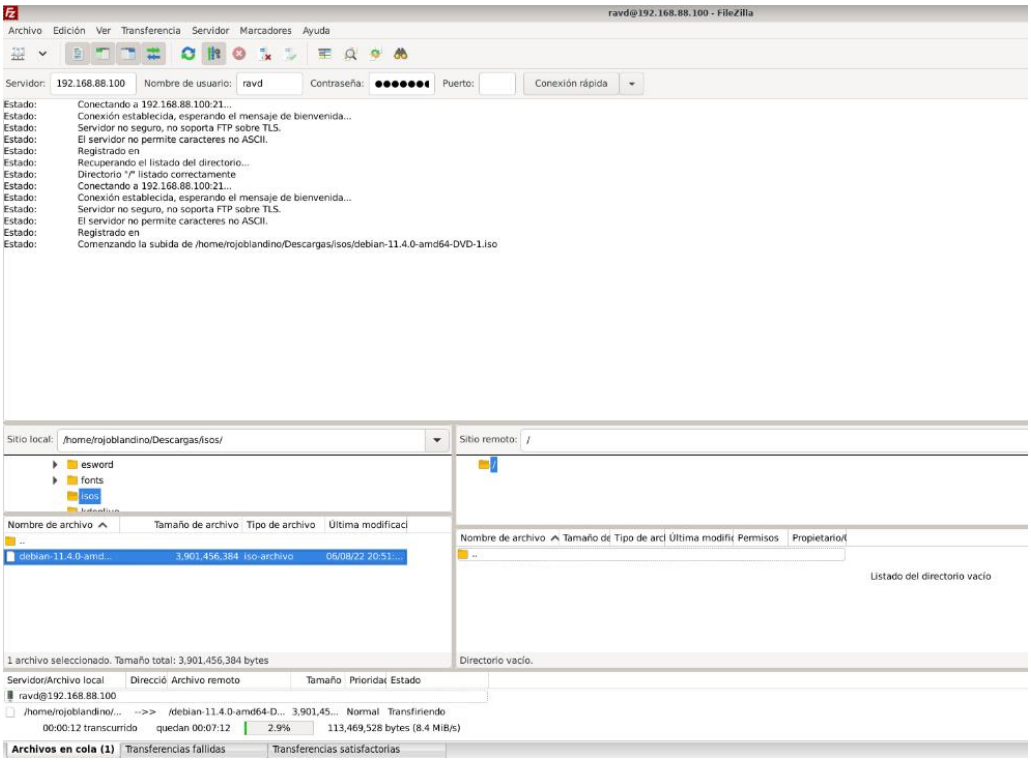


Figura (35) - Inicio de subida del archivo

Para la prueba se detuvo nbdc1 completamente para simular el fallo del host y comprobar el paso de los recursos y adquisición de los mismos por nbdc2

Cuadro (088) - Revisión de recursos en nbdc2

```
root@nbdc2:~# df -h
S.ficheros           Tamaño Usados  Disp Uso% Montado en
/dev/sda1              7.4G   1.3G  5.8G  18% /
udev                    10M      0   10M   0% /dev
tmpfs                   49M   4.5M   44M  10% /run
tmpfs                  122M      0  122M   0% /dev/shm
tmpfs                  5.0M      0  5.0M   0% /run/lock
tmpfs                  122M      0  122M   0% /sys/fs/cgroup
/dev/mapper/VGA-vwh     22G   452M   22G   3% /srv/disk
```

Una vez comprobado que los recursos existen en nbdc2 se realizó la conexión a este nuevo controlador:

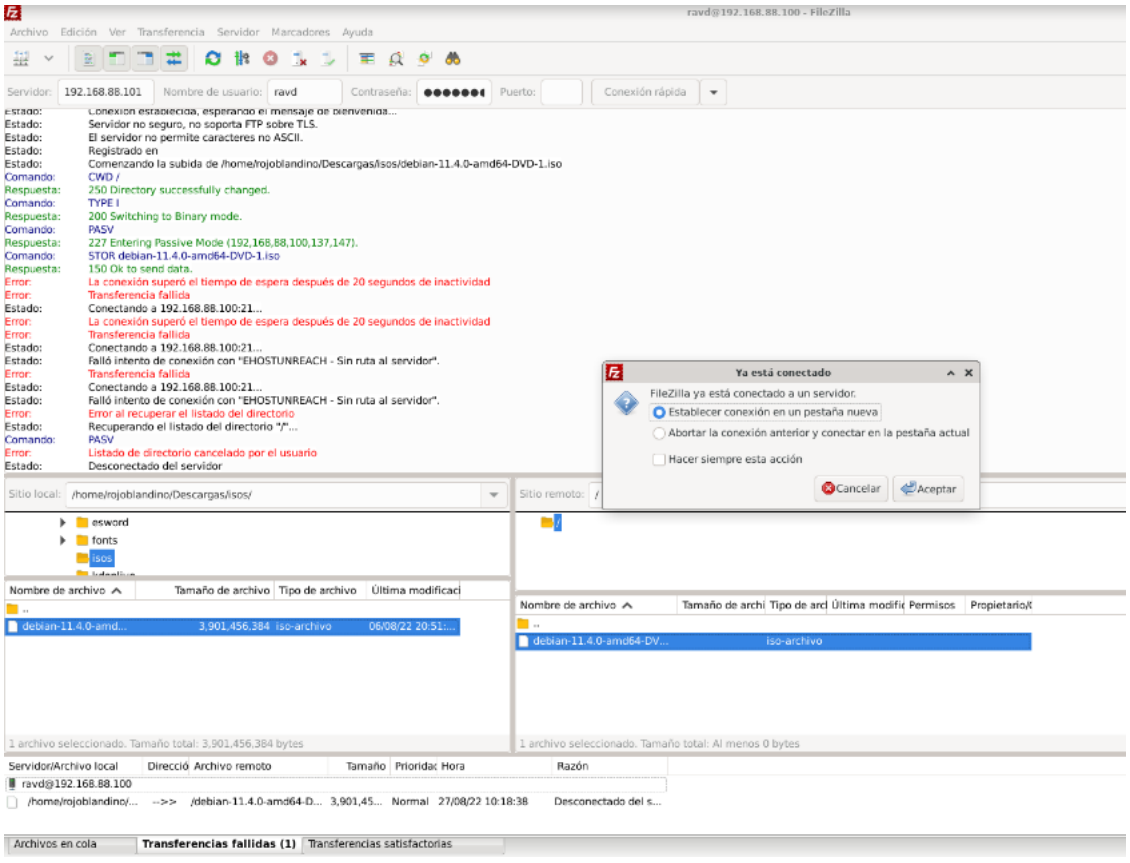


Figura (36) - Inicio de sesión en el controlador NBD esclavo que fue promovido a maestro

Al establecer la sesión FTP se encontró el archivo que se inició a transferir antes del fallo, observando que el tamaño de este estaba incompleto.

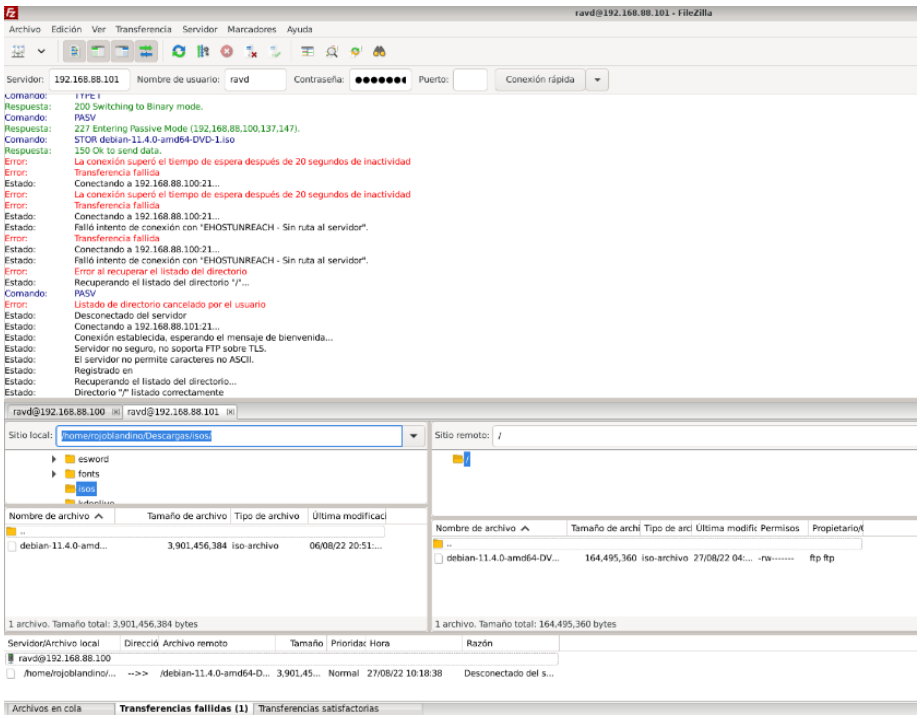


Figura (37) - Se observa archivo y tamaño incompleto

Se seleccionó el archivo nuevamente, y se optó por elegir la opción reanudar para que el archivo continúe en el punto donde había quedado anteriormente.

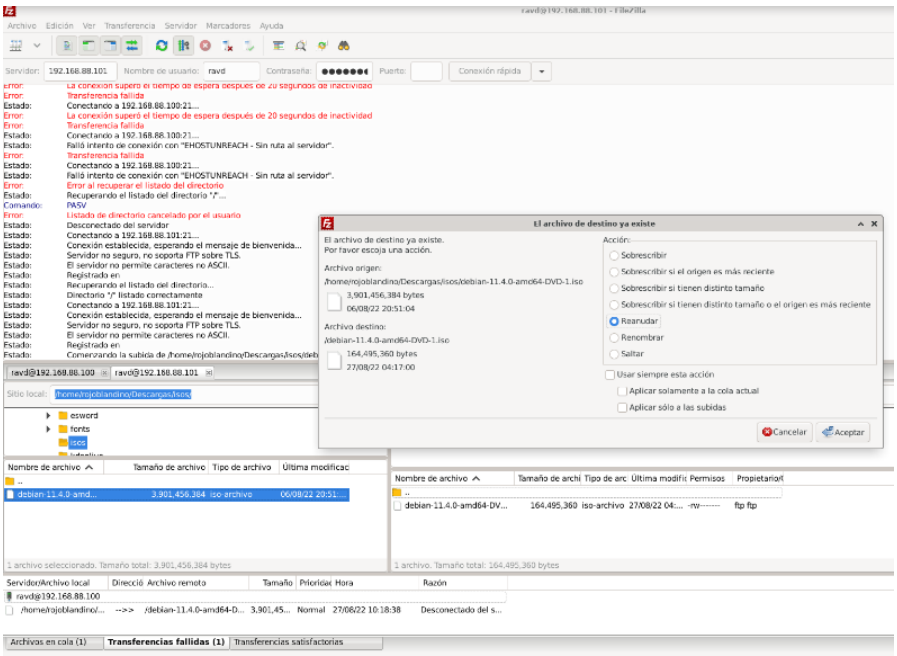


Figura (38) - Subida nuevamente del archivo, escogiendo reanudar proceso

El archivo continúo transfiriéndose hasta terminar de completarse.

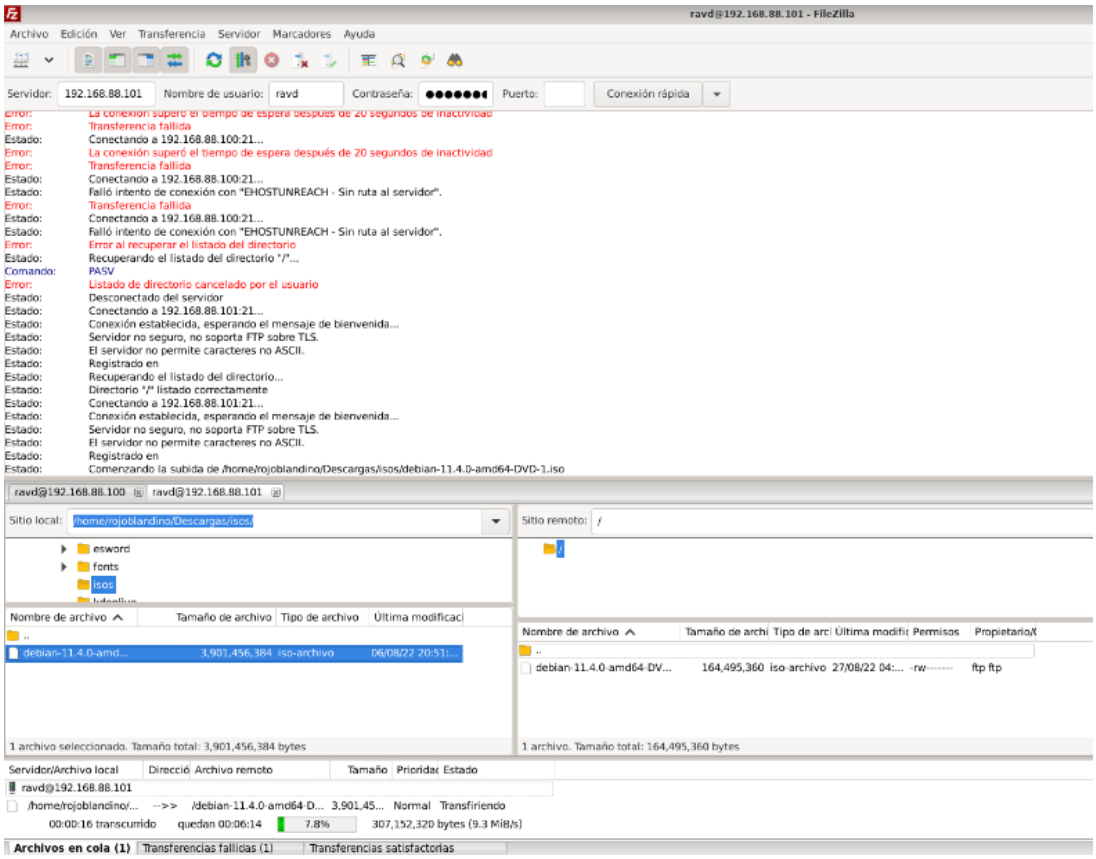


Figura (39) - Reanudando subida de archivo donde quedó al momento del fallo

V.2.1.4.Agregar un nuevo miembro al clúster RAVD

La siguiente prueba tuvo como objetivo mostrar los pasos para agregar un nuevo miembro al clúster existente.

Para agrandar el arreglo, el primer paso fue agregar un nuevo **SPN**, el cual está conformado por un **MA-SVN** y un **SL-SVN**, en cada una de estas dos máquinas virtuales se le instaló un Sistema Operativo, y todas las paqueterías explicadas en el capítulo IV.

Se consultó el tamaño actual del arreglo para tener el punto de partida con el cual compararlo una vez que el nuevo miembro este agregado:

Cuadro (089) - Consulta de las particiones del sistema

```
root@nbdc1:~# df -h
S.ficheros          Tamaño Usados  Disp Uso% Montado en
/dev/sda1             7.4G   1.5G  5.6G  21% /
udev                   10M      0   10M   0% /dev
tmpfs                  46M   4.8M   41M  11% /run
tmpfs                 115M      0  115M   0% /dev/shm
tmpfs                 5.0M      0  5.0M   0% /run/lock
tmpfs                 115M      0  115M   0% /sys/fs/cgroup
/dev/mapper/VGA-vwh    22G    11G   12G  46% /srv/disk
```

Luego se consultaron los detalles del arreglo:

Cuadro (090) - Revisión de los detalles arreglo

```
root@nbdc1:~# mdadm -D /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Thu Aug 18 11:03:27 2022
     Raid Level : raid6
     Array Size : 23049216 (21.98 GiB 23.60 GB)
  Used Dev Size : 11524608 (10.99 GiB 11.80 GB)
   Raid Devices : 4
  Total Devices : 6
    Persistence : Superblock is persistent

    Update Time : Thu Aug 18 16:51:28 2022
          State : clean
 Active Devices : 4
Working Devices : 6
 Failed Devices : 0
  Spare Devices : 2

         Layout : left-symmetric
     Chunk Size : 512K

           Name : nbdc1:0  (local to host nbdc1)
           UUID : b4019de9:0ffc2bb5:24a98a51:0e5433ef
         Events : 33

    Number   Major   Minor   RaidDevice State
       0      43        0        0      active sync   /dev/nbd0
       1      43       16        1      active sync   /dev/nbd1
       2      43       32        2      active sync   /dev/nbd2
       3      43       48        3      active sync   /dev/nbd3

       4      43       64        -      spare   /dev/nbd4
       5      43       80        -      spare   /dev/nbd5
```

Se agregó el nuevo disco en el servicio nbd y se reinician los servicios:

Cuadro (091) - Adición del dispositivo de bloques al arreglo

```
root@nbdc1:~# cat << EOF >> /etc/nbd-client

NBD_DEVICE[6]=/dev/nbd6
NBD_TYPE[6]="r"
NBD_HOST[6]="172.16.2.56"
NBD_PORT[6]=10809
NBD_NAME[6]="disk7"
NBD_EXTRA[6]=
EOF
root@nbdc1:~# systemctl restart heartbeat
```

Al configurar el nuevo **SPN**, se obtuvo un nuevo dispositivo de bloques **nbd**, el cual fue agregado al arreglo mediante el siguiente comando:

Cuadro (092) - Adición del dispositivo de bloques al arreglo

```
root@nbdc1:~# mdadm /dev/md0 --add /dev/nbd6
mdadm: added /dev/nbd6
```

Luego de agregado el disco, se observó que el nuevo miembro fue marcado como spare:

Cuadro (093) - Revisión de los detalles arreglo

```
root@nbdc1:~# mdadm -D /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Thu Aug 18 11:03:27 2022
     Raid Level : raid6
     Array Size : 23049216 (21.98 GiB 23.60 GB)
  Used Dev Size : 11524608 (10.99 GiB 11.80 GB)
   Raid Devices : 4
  Total Devices : 7
    Persistence : Superblock is persistent

    Update Time : Thu Aug 18 16:59:39 2022
          State : clean
 Active Devices : 4
Working Devices : 7
 Failed Devices : 0
  Spare Devices : 3

         Layout : left-symmetric
     Chunk Size : 512K

           Name : nbdc1:0  (local to host nbdc1)
           UUID : b4019de9:0ffc2bb5:24a98a51:0e5433ef
         Events : 36

    Number   Major   Minor   RaidDevice State
       0      43        0        0      active sync   /dev/nbd0
       1      43       16        1      active sync   /dev/nbd1
```

```
       2       43       32        2      active sync   /dev/nbd2
       3       43       48        3      active sync   /dev/nbd3

       4       43       64        -      spare   /dev/nbd4
       5       43       80        -      spare   /dev/nbd5
       6       43       96        -      spare   /dev/nbd6
```

Mediante la opción "**grow**" para el comando "**mdadm**", se indicó que el arreglo crecería y adicionaría el espacio libre, que en este caso fue el disco nuevo en modalidad **SPARE**; otra opción adicional importante configurada fue "**backup-file**", quien se encarga de almacenar un respaldo del arreglo, para que, en caso de fallos durante el agrandamiento, haga posible la reconstrucción del arreglo nuevamente:

Cuadro (094) - Agrandamiento del arreglo

```
root@nbdc1:~#     mdadm     --grow     --raid-devices=5     --backup-
file=/root/md0_grow.bak /dev/md0
mdadm: Need to backup 3072K of critical section..
```

Luego se consultó nuevamente el arreglo, observando que el estado del mismo pasó a tener los valores "**clean, reshaping**", y que el proceso de agrandamiento estaba a un 55% de su totalidad, motivo por el cual, a pesar de que el nuevo disco ya se reconoce como miembro activo, el tamaño del arreglo permanece sin cambios, ya que la nueva capacidad de almacenamiento se verá reflejada hasta que el proceso de agrandamiento finalice en su 100%:

Cuadro (095) - Revisión de los detalles arreglo

```
root@nbdc1:~# mdadm -D /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Thu Aug 18 11:03:27 2022
     Raid Level : raid6
     Array Size : 23049216 (21.98 GiB 23.60 GB)
  Used Dev Size : 11524608 (10.99 GiB 11.80 GB)
   Raid Devices : 5
  Total Devices : 7
    Persistence : Superblock is persistent

    Update Time : Thu Aug 18 17:01:29 2022
          State : clean, reshaping
 Active Devices : 5
Working Devices : 7
 Failed Devices : 0
  Spare Devices : 2

         Layout : left-symmetric
     Chunk Size : 512K
```

```
 Reshape Status : 55% complete
  Delta Devices : 1, (4->5)

           Name : nbdc1:0  (local to host nbdc1)
           UUID : b4019de9:0ffc2bb5:24a98a51:0e5433ef
         Events : 50

    Number   Major   Minor   RaidDevice State
       0      43        0        0      active sync   /dev/nbd0
       1      43       16        1      active sync   /dev/nbd1
       2      43       32        2      active sync   /dev/nbd2
       3      43       48        3      active sync   /dev/nbd3
       6      43       96        4      active sync   /dev/nbd6

       4      43       64        -      spare   /dev/nbd4
       5      43       80        -      spare   /dev/nbd5
```

Posteriormente se consultó el tiempo estimado faltante:

Cuadro (096) - Comprobación y monitoreo del estado del arreglo

```
root@nbdc1:~# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid6 nbd6[6] nbd0[0] nbd4[4](S) nbd5[5](S) nbd3[3] nbd2[2]
nbd1[1]
      23049216 blocks super 1.2 level 6, 512k chunk, algorithm 2 [5/5]
[UUUUU]
      [>....................]    reshape =    2.3%  (265860/11524608)
finish=26.9min speed=6952K/sec

unused devices: <none>
root@nbdc1:~# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid6 nbd6[6] nbd0[0] nbd4[4](S) nbd5[5](S) nbd3[3] nbd2[2]
nbd1[1]
      23049216 blocks super 1.2 level 6, 512k chunk, algorithm 2 [5/5]
[UUUUU]
      [==========>.........]    reshape =  50.0%  (5764076/11524608)
finish=4.4min speed=21637K/sec

unused devices: <none>
root@nbdc1:~# cat /proc/mdstat
Personalities : [raid6] [raid5] [raid4]
md0 : active raid6 nbd6[6] nbd0[0] nbd4[4](S) nbd5[5](S) nbd3[3] nbd2[2]
nbd1[1]
      34573824 blocks super 1.2 level 6, 512k chunk, algorithm 2 [5/5]
[UUUUU]

unused devices: <none>
```

Terminado el proceso se consultó nuevamente los detalles del arreglo, logrando observar dos cambios; el primer cambio fue que el estado paso a valor “**clean**” y el

segundo, es que el tamaño del arreglo fue actualizado a su nueva capacidad de almacenamiento:

Cuadro (097) - Revisión de los detalles arreglo

```
root@nbdc1:~# mdadm -D /dev/md0
/dev/md0:
        Version : 1.2
  Creation Time : Thu Aug 18 11:03:27 2022
     Raid Level : raid6
     Array Size : 34573824 (32.97 GiB 35.40 GB)
  Used Dev Size : 11524608 (10.99 GiB 11.80 GB)
   Raid Devices : 5
  Total Devices : 7
    Persistence : Superblock is persistent

    Update Time : Thu Aug 18 17:09:20 2022
          State : clean
 Active Devices : 5
Working Devices : 7
 Failed Devices : 0
  Spare Devices : 2

         Layout : left-symmetric
     Chunk Size : 512K

           Name : nbdc1:0  (local to host nbdc1)
           UUID : b4019de9:0ffc2bb5:24a98a51:0e5433ef
         Events : 103

    Number   Major   Minor   RaidDevice State
       0      43        0        0      active sync   /dev/nbd0
       1      43       16        1      active sync   /dev/nbd1
       2      43       32        2      active sync   /dev/nbd2
       3      43       48        3      active sync   /dev/nbd3
       6      43       96        4      active sync   /dev/nbd6

       4      43       64        -      spare   /dev/nbd4
       5      43       80        -      spare   /dev/nbd5
```

Sin embargo, habiendo agrandado el arreglo no es suficiente para hacer uso de su nueva capacidad de almacenamiento, ya que el volumen físico del LVM mantiene el tamaño original, dejando el nuevo espacio agregado sin ser reclamado, por lo tanto, es necesario indicarle al volumen físico que lo haga.

Cuadro (098) - Montaje del volumen lógico y listado de particiones

```
root@nbdc1:~# mount /dev/VGA/vwh /srv/disk/
root@nbdc1:~# df -h
S.ficheros          Tamaño Usados  Disp Uso% Montado en
/dev/sda1             7.4G   1.5G  5.6G  21% /
udev                   10M      0   10M   0% /dev
```

```
tmpfs                    46M   4.8M   41M  11% /run
tmpfs                   115M      0  115M   0% /dev/shm
tmpfs                   5.0M      0  5.0M   0% /run/lock
tmpfs                   115M      0  115M   0% /sys/fs/cgroup
/dev/mapper/VGA-vwh      22G     11G   12G  46% /srv/disk
```

Se reclamó el espacio libre agregándoselo al volumen físico mediante el comando "**pvresize**" y luego al volumen lógico mediante el comando "**lvextend**", permitiendo de esta forma que el **XFS** pueda hacer uso completo de la capacidad de almacenamiento total del arreglo:

Cuadro (099) - Redimensionamiento del volumen físico y extensión del volumen lógico

```
root@nbdc1:~# pvresize /dev/md0
  Physical volume "/dev/md0" changed
  1 physical volume(s) resized / 0 physical volume(s) not resized
root@nbdc1:~# lvextend -l +100%FREE /dev/mapper/VGA-vwh
  Size of logical volume VGA/vwh changed from 21.98 GiB (5627 extents)
to 32.97 GiB (8440 extents).
  Logical volume vwh successfully resized
```

El último paso fue hacer crecer el sistema de archivos XFS, reclamando todo el espacio libre que tenía disponible:

Cuadro (100) - Redimensionando el sistema de fichero XFS

```
root@nbdc1:~# xfs_growfs /dev/mapper/VGA-vwh
meta-data=/dev/mapper/VGA-vwh    isize=256    agcount=16, agsize=360064
blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                      bsize=4096   blocks=5761024, imaxpct=25
         =                       sunit=128    swidth=256 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal               bsize=4096   blocks=2816, version=2
         =                      sectsz=512   sunit=8 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 5761024 to 8642560
```

Se consultó el tamaño de las particiones para corroborar que los cambios del tamaño en la partición fueron efectivos:

Cuadro (101) - Listado de particiones

```
root@nbdc1:~# df -h
S.ficheros          Tamaño Usados  Disp Uso% Montado en
/dev/sda1             7.4G   1.5G  5.6G  21% /
udev                   10M      0   10M   0% /dev
```

```
tmpfs                    46M   4.8M   41M  11% /run
tmpfs                   115M      0  115M   0% /dev/shm
tmpfs                   5.0M      0  5.0M   0% /run/lock
tmpfs                   115M      0  115M   0% /sys/fs/cgroup
/dev/mapper/VGA-vwh      33G     11G   23G  31% /srv/disk
```

V.2.2. Rendimiento

V.2.2.1. Comparación de tiempo de escritura/lectura en disco para el modelo RAVD con diferentes capacidades de Memoria Interna

Para realizar la prueba de rendimiento se utilizó el comando **time** para calcular el tiempo realizado al hacer el benchmark del rendimiento del uso del disco en red para calcular el **I/O** del disco en red al copiar los datos.

La definición oficial según la **man page** del comando time es la siguiente:

Cuadro (102) - Comando time

```
       La  orden  time  ejecuta el programa orden con los argumentos
suministrados. Cuando orden finaliza, time escribe un mensaje en la
salida estándar devolviendo estadísticas temporales sobre la ejecución
de este programa.  Estas estadísticas están compuestas por (i) el tiempo
real transcurrido entre la llamada y la finalización de orden , (ii) el
tiempo de  usuario  del  procesador (la  suma  de los valores tms_utime
y tms_cutime en un struct tms tal y como devuelve times(2)), y (iii) el
tiempo de sistema del procesador (la suma de los valores tms_stime y
tms_cstime en un struct tms tal y como devuelve times(2)).
```

El comando **time** devuelve tres resultados por defecto, los cuales se explican a continuación:

- **Real:** Tiempo real transcurrido desde que el proceso inició hasta que terminó (reloj de pared), el tiempo está dado en [horas:] minutos: segundos.
- **User:** Número total de **CPU** que el proceso utilizó directamente (en modo de usuario), la unidad de medida está dada en segundos.
- **Sys:** Número total de **CPU** utilizado por el sistema en favor del proceso (en modo kernel), la unidad de medida está dada en segundos.

Se realizaron varias pruebas usando 3 capacidades de Memoria Interna diferentes, en el servidor **controlador** del nodo, en cada capacidad se realizaron 3 pruebas comparativas, las tres capacidades usadas para la prueba fueron:

- Primera prueba con 512M de Memoria Interna.
- Segunda prueba con 1GB de Memoria Interna.
- Tercera prueba con 2GB de Memoria Interna.

Pruebas realizadas con 512M de Memoria Interna los resultados fueron los siguientes:

Cuadro (103) - Revisando memoria

```
root@nbdc1:/srv/disk# free -m
             total       used       free     shared    buffers     cached
Mem:           494        130        363          4          9         44
-/+ buffers/cache:         77        416
Swap:         1905          0       1905
```

Escribiendo un archivo de 11GB:

Cuadro (104) - Primera prueba creación de archivo de 11GB

```
root@nbdc1:~# time dd if=/dev/zero of=/srv/disk/a.dd bs=1M count=10240
10240+0 registros leídos
10240+0 registros escritos
10737418240 bytes (11 GB) copiados, 210.378 s, 51.0 MB/s

real    3m30.422s
user    0m0.092s
sys     0m8.164s
```

Cuadro (105) - Segunda prueba creación de archivo de 11GB

```
root@nbdc1:~# time dd if=/dev/zero of=/srv/disk/a.dd bs=1M count=10240
10240+0 registros leídos
10240+0 registros escritos
10737418240 bytes (11 GB) copiados, 210.391 s, 51.0 MB/s

real    3m30.445s
user    0m0.092s
sys     0m8.000s
```

Cuadro (106) - Tercera prueba creación de archivo de 11GB

```
root@nbdc1:~# time dd if=/dev/zero of=/srv/disk/a.dd bs=1M count=10240
10240+0 registros leídos
10240+0 registros escritos
```

```
10737418240 bytes (11 GB) copiados, 209.205 s, 51.3 MB/s

real    3m29.291s
user    0m0.052s
sys     0m8.500s
```

Pruebas realizadas con 1GB de Memoria Interna los resultados fueron los siguientes:

Cuadro (107) - Revisando memoria

```
root@nbdc1:~# free -m
             total       used       free     shared    buffers     cached
Mem:          1000        131        868          4          8         44
-/+ buffers/cache:         78        921
Swap:         1905          0       1905
```

Escribiendo un archivo de 11GB:

Cuadro (108) - Primera prueba creación de archivo de 11G

```
root@nbdc1:~# time dd if=/dev/zero of=/srv/disk/a.dd bs=1M count=10240
10240+0 registros leídos
10240+0 registros escritos
10737418240 bytes (11 GB) copiados, 207.821 s, 51.7 MB/s

real    3m27.896s
user    0m0.052s
sys     0m11.708s
```

Cuadro (109) - Segunda prueba creación de archivo de 11GB

```
root@nbdc1:~# time dd if=/dev/zero of=/srv/disk/a.dd bs=1M count=10240
10240+0 registros leídos
10240+0 registros escritos
10737418240 bytes (11 GB) copiados, 208.692 s, 51.5 MB/s

real    3m28.772s
user    0m0.088s
sys     0m7.968s
```

Cuadro (110) - Tercera prueba creación de archivo de 11GB

```
root@nbdc1:~# time dd if=/dev/zero of=/srv/disk/a.dd bs=1M count=10240
10240+0 registros leídos
10240+0 registros escritos
10737418240 bytes (11 GB) copiados, 210.063 s, 51.1 MB/s

real    3m30.189s
user    0m0.092s
sys     0m8.668s
```

Pruebas realizadas con 2GB de Memoria Interna los resultados fueron los siguientes:

Cuadro (111) - Revisando memoria

```
root@nbdc1:~# free -m
             total       used       free     shared    buffers     cached
Mem:          2010        134       1875          5          8         45
-/+ buffers/cache:         80       1930
Swap:         1905          0       1905
```

Escribiendo un archivo de 11GB:

Cuadro (112) - Primera prueba creación de archivo de 11G

```
root@nbdc1:~# time dd if=/dev/zero of=/srv/disk/a.dd bs=1M count=10240
10240+0 registros leídos
10240+0 registros escritos
10737418240 bytes (11 GB) copiados, 111.076 s, 96.7 MB/s

real    1m51.409s
user    0m0.032s
sys     0m8.396s
```

Cuadro (113) - Segunda prueba creación de archivo de 11GB

```
root@nbdc1:~# time dd if=/dev/zero of=/srv/disk/a.dd bs=1M count=10240
10240+0 registros leídos
10240+0 registros escritos
10737418240 bytes (11 GB) copiados, 110.387 s, 97.3 MB/s

real    1m50.605s
user    0m0.012s
sys     0m8.876s
```

Cuadro (114) - Tercera prueba creación de archivo de 11GB

```
root@nbdc1:~# time dd if=/dev/zero of=/srv/disk/a.dd bs=1M count=10240
10240+0 registros leídos
10240+0 registros escritos
10737418240 bytes (11 GB) copiados, 109.289 s, 98.2 MB/s

real    1m49.499s
user    0m0.056s
sys     0m8.496s
```

Descripción del resultado:

Con las pruebas realizadas bajo el mismo ambiente, pero con capacidades de Memoria Interna diferentes, se logró observar que los tiempos de respuesta son inversamente proporcional a los recursos de los equipos, es decir que mientras más Memoria Interna tenga el modelo, los tiempos de respuesta para lectura/escritura en el arreglo de discos virtuales, disminuirán considerablemente.

Luego de haber realizado las pruebas, y presentado el prototipo a REKASA, la gerente general de dicha empresa nos dio por aprobado el modelo propuesto. Para ver carta de aprobación favor dirigirse al APÉNDICE K.

CAPITULO VI.CONCLUSIONES Y RECOMENDACIONES

VI.1. Conclusiones

1. Se diseñó exitosamente un Sistema de Respaldo de Datos en Línea que satisface los requerimientos definidos, proveyendo una solución efectiva al problema de almacenamiento de datos de REKASA, a partir de la integración de tecnologías open source existentes, y cuya efectividad fue comprobada a través de la implementación y evaluación de un prototipo funcional, con resultados satisfactorios.

2. El resultado de las evaluaciones permitió demostrar que el modelo RAVD desarrollado, implementado en el prototipo funcional evaluado, brinda las siguientes bondades que satisfacen los requerimientos de REKASA:
    a. **Bajo costo:** se redujo cuantiosamente el costo económico de la solución, por cuanto el modelo permite:
        - Utilizar software y licencias sin costo alguno.
        - Utilizar hardware existente en desuso por REKASA.
    b. **Estabilidad:** el sistema se mantuvo en funcionamiento ante los eventos de falla provocados de pérdida de un miembro del clúster en frío o en caliente, siéndose imperceptible dichos fallos para el servicio al momento de estar en un proceso de escritura o lectura al arreglo de discos virtuales.
    c. **Disponibilidad:** tanto el sistema como los datos estuvieron disponibles para el usuario en todo momento ante los eventos de falla que se provocaron, manteniendo de esta forma el servicio y los datos en línea.
    d. **Integridad de datos:** se verificó que los datos almacenados durante el copiado no se perdieron cuando se simularon fallos o reinicios de las máquinas que estaban manteniendo los datos en línea.

e. **Escalabilidad:** se verificó que el modelo es escalable, pudiéndosele adicionar nuevos discos al arreglo en caliente (de acuerdo con los recursos disponibles) y estos discos pueden quedar en modalidad spare (con finalidad de backup inmediato) o como miembros activos del arreglo.

f. Rendimiento: se verificó que el modelo presenta buenos tiempos de respuesta, los cuales son aceptables conforme lo esperado por REKASA, y para la aplicación que ellos requieren. En las pruebas se pudo identificar que los tiempos de respuesta son inversamente proporcional a los recursos de los equipos, por lo que, si se desea mayor desempeño y rendimiento, se puede obtener con el aumento de los recursos, principalmente de Memoria Interna, en los equipos controladores.

3. Las técnicas y tecnología de software libre seleccionadas resultaron idóneas para implementar un clúster en arreglo redundante de discos virtuales, que fue la base para el diseño del sistema de respaldo, lográndose hacerlas trabajar de una manera orquestada, satisfaciendo la necesidad de la empresa. Por lo antes mencionado, se considera que dicho modelo puede ser usado para fines educativos, para lo cual el presente documento representa una guía bien detallada.

4. El prototipo funcional del sistema de respaldo en línea basado en un clúster de almacenamiento en arreglo redundante de discos virtuales (RAVD) implementado, representó de forma funcionalmente equivalente y arquitecturalmente correcta el modelo RAVD, y permitió validar, a partir de pruebas de ensayo, el correcto funcionamiento, estabilidad, escalabilidad, alta disponibilidad, y buen desempeño del mismo, corroborándose que es una solución efectiva, y adecuada a las necesidades de REKASA.

VI.2. Recomendaciones

A partir de los resultados de la evaluación del funcionamiento del sistema de respaldo en línea basado en clúster de almacenamiento en Arreglo Redundante de Discos Virtuales RAVD, se consideran pertinentes las siguientes recomendaciones:

1. En virtud de que RAID6 soporta la caída de hasta dos discos virtuales del arreglo simultáneamente sin pérdida de datos, se recomienda que, en la implementación de este modelo, se invierta al menos en dos discos de reemplazo inmediato (SPARE), con la finalidad de aumentar la capacidad de pérdidas de discos en caliente.
2. En vista de que el modelo integra diferentes tecnologías para generar una solución innovadora de bajo costo para servicio de almacenamiento en red, se recomienda replicarla con fines educativos.
3. Si se decide replicar este modelo con fines educativos, según lo explicado en la sección IV.4.2. Realización de la instalación del Sistema Operativo, recomendamos clonar las máquinas sin los paquetes DRBD y HEARTBEAT (*el cual no podría vincularse con el mismo, ya que al clonarse no reconocería a su dispositivo par como un dispositivo diferente evitando consultar el estado de salud entre ambos*), ya que estos deben de ser instalados manualmente en cada una de las máquinas luego de haber sido clonadas
4. En caso de hacer uso del virtualizador PROXMOX para fines educativos utilizando contenedores LXC, no se recomienda crear los contenedores en modo “no privilegiado”, porque no tendría privilegio de root para leer dispositivos de bloques. Si se desea usar PROXMOX como hipervisor para ambiente productivo, se recomienda usar KVM y no LXC.
5. Para un ambiente productivo donde se quisiera mejorar el rendimiento, se recomienda incrementar los recursos físicos, principalmente la Memoria Interna, dado que en las pruebas se hizo manifiesto que la cantidad de Memoria Interna es inversamente proporcional a los tiempos

de respuesta de lectura/escritura de datos en los discos virtuales, así como también en la transmisión de datos en red.

6. Se recomienda que los equipos físicos tengan como mínimo dos tarjetas de red para separar físicamente el transporte de datos de monitoreo como de los datos de servicio, para mejor desempeño y buena salud del clúster.
7. Se recomienda implementar un sistema de monitoreo mediante soluciones de software libre como Nagios, Cacti o Zabbix, o mediante software comercial como Orion.

BIBLIOGRAFÍA[54]

Arjuna, C. (28 de Agosto de 2013). *HPCC Systems WIKI*. Recuperado el 15 de enero de 2022, de Wiki HPCC Systems: https://wiki.hpccsystems.com/display/hpcc/Home

Brush, K., & Kirsch, B. (Diciembre de 2010). *Virtualization*. Recuperado el 30 de enero de 2022, de TechTarget: http://searchservervirtualization.techtarget.com/definition/virtualization

Chidlow, S. (2003). *JISC Technology and Standards Watch Report: Storage Area Networks.*

CIFS. (2015). Recuperado el 11 de enero de 2022, de Advancing storage and information technology: https://www.snia.org/education/online-dictionary/term/cifs

Cluster. (2015). Recuperado el 27 de enero de 2022, de Advancing storage and information technology: https://www.snia.org/education/online-dictionary/term/cluster

Common RAID Disk Data Format Specification. (27 de marzo de 2019). Recuperado el 5 de febrero de 2022, de Advancing storage and information technology: http://www.snia.org/sites/default/files/SNIA_DDF_Technical_Position_v2.0.pdf

Computer Hope. (Noviembre de 2018). *Floating Point Operations Per Second (FLOPS)*. Recuperado el 24 de febrero de 2022, de Computer Hope, Free

[54] **Referencias Bibliográficas:** las referencias bibliográficas fueron estructuradas conforme norma APA sexta edición, empleando la herramienta de gestión de citas de MS Word.

Computer help and information: http://www.computerhope.com/jargon/f/flops.htm

DAS. (15 de 10 de 2015). Recuperado el 15 de marzo de 2022, de Advancing storage and information technology: https://www.snia.org/education/online-dictionary/term/direct-attached-storage

Eastlake 3rd, D., & Jones, P. (Septiembre de 2001). *"US Secure Hash Algorithm 1 (SHA1)", RFC 3174*. doi:10.17487/RFC3174

Failover. (2015). Recuperado el 23 de marzo de 2022, de Advancing storage and information technology: https://www.snia.org/education/online-dictionary/term/failover

Fundación FreeBSD. (2016). *FreeBSD*. Recuperado el 28 de enero de 2022, de FreeBSD: http://www.freebsd.org/es/

Gleeson, B., Lin, A., Heinanen, J., Armitage, G., & Malis, A. (Febrero de 2000). *"A Framework for IP Based Virtual Private Networks", RFC 2764*. doi:10.17487/RFC2764

Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *SCIENCE*, 60-65.

HLINBIT HA-Solutions GmbH [SplitBrain]. (29 de Enero de 2010). *HLINBIT HA-Solutions GmbH, The Linux-HA Project*. Recuperado el 10 de febrero de 2022, de HLINBIT HA-Solutions GmbH, The Linux-HA Project: http://www.linux-ha.org/wiki/Split_Brain

ISO/IEC 7498-1. (1996). International Standard - Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model. En ISO/IEC, *International Standard* (pág. 37). Suiza: ISO/IEC. Obtenido de http://www.ecma-international.org/activities/Communications/TG11/s020269e.pdf

LINBIT HA-Solutions GmbH [Linux-HA]. (13 de December de 2010). *LINBIT HA-Solutions GmbH, The Linux-HA Project*. Recuperado el 20 de febrero de 2022, de http://linux-ha.org/wiki/Main_Page

Linux-VServer. (6 de Junio de 2013). *Linux-VServer*. Recuperado el 28 de enero de 2022, de Linux-VServer: http://linux-vserver.org/Welcome_to_Linux-VServer.org

Lutkevich, B. (Febrero de 2015). *Framework*. Recuperado el 28 de enero de 2022, de Techtarget: http://whatis.techtarget.com/definition/framework

Miller, L., & Fadden, S. (2014). *Software Defined Storage For Dummies.* John Wiley & Sons.

NAS. (5 de 10 de 2015). Recuperado el 27 de enero de 2022, de Advancing storage and information technology: https://www.snia.org/education/online-dictionary/term/network-attached-storage

Nowicki, B. (Marzo de 1989). *"NFS: Network File System Protocol specification", RFC 1094*. doi:10.17487/RFC1094

Parallels IP Holdings GmbH. (2015). *Virtuozzo*. Recuperado el 17 de febrero de 2022, de Virtuozzo: http://www.virtuozzo.com/

Paredes, J. P. (22 de Noviembre de 2001). *Alta disponibilidad para Linux.* Recuperado el 20 de febrero de 2022, de http://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200103hispalinux/paredes/pdf/LinuxHA.pdf

Paul Kirvan. (Abril de 2006). *Clusters*. Recuperado el 18 de marzo de 2022, de Techtarget: http://searchexchange.techtarget.com/definition/cluster

Poelker, C., & Nikitin, A. (2014). *Storage Area Networks For Dummies (2da ed.).*

Postel, J. (28 de Agosto de 1980). *"User Datagram Protocol", STD 6, RFC 768*. doi:10.17487/RFC0768

Postel, J., & Reynolds, J. (Octubre de 1985). *"File Transfer Protocol", STD 9, RFC 959*. doi:10.17487/RFC0959

Rivest, R. (Abril de 1992). *"The MD5 Message-Digest Algorithm", RFC 1321*. doi:10.17487/RFC1321

SAN. (2015). Recuperado el 14 de enero de 2022, de Advancing storage and information technology: https://www.snia.org/education/online-dictionary/term/san

Socolofsky, T., & Kale, C. (Enero de 1991). *"TCP/IP tutorial", RFC 1180*. doi:10.17487/RFC1180

Softpanorama. (s.f.). *High Performance Computing (HPC)*. Recuperado el 8 de marzo de 2022, de Softpanorama: https://softpanorama.org/HPC/index.shtml

Sollins, K. (Julio de 1992). *"The TFTP Protocol (Revision 2)", STD 33, RFC 1350*. doi:10.17487/RFC1350

Suppi Boldrito, R. (2015). *Clusterización.* Recuperado el 20 de febrero de 2022, de UOC (Universitat Oberta de Catalunya): https://www.exabyteinformatica.com/uoc/Informatica/Administracion_avanzada_del_sistema_operativo_GNU_Linux/Administracion_avanzada_del_sistema_operativo_GNU_Linux_%28Modulo_6%29.pdf

SwitchOver. (s.f.). Recuperado el 10 de febrero de 2022, de SIOS Technology Corp.: https://docs.us.sios.com/spslinux/9.5.1/en/topic/switch-to-standby-node-to-confirm-switchover-is-working

TakeOver. (s.f.). Recuperado el 10 de febrero de 2022, de SIOS Technology Corp.: https://docs.us.sios.com/spslinux/9.6.0/en/topic/what-is-quot-split-brain-quot-and-how-to-avoid-it

Tanenbaum, A. (2009). Dispositivos de E/S. En A. Tanenbaum, *Sistemas Operativos Modernos (3ra ed.)* (págs. 27-30). México: Pearson Educación.

Tanenbaum, A. (2009). Fork. En A. Tanenbaum, *Sistemas Operativos Modernos (3ra ed.)* (pág. 61). México: Pearson Educación.

Tate, J., Bernasconi, A., Mescher, P., & Scholten, F. (2003). *Introduction to Storage.*

TechTerms. (22 de Enero de 2009). *Floating Point Operations Per Second (FLOPS)*. Recuperado el 24 de febrero de 2022, de TechTerms: http://techterms.com/definition/flops

Virtual Box. (2016). Recuperado el 26 de enero de 2022, de Virtual Box: https://www.virtualbox.org/

Virtuozzo. (2016). *OpenVZ*. Recuperado el 28 de enero de 2022, de OpenVZ: https://openvz.org/

VMWARE ESXI. (2016). Recuperado el 12 de enero de 2022, de VMWARE: http://www.vmware.com/products/vsphere-hypervisor/

Webopedia. (7 de enero de 1998). *Floating Point Operations Per Second (FLOPS)*. Recuperado el 24 de febrero de 2022, de Webopedia: http://www.webopedia.com/TERM/F/FLOPS.html

XEN. (2016). Recuperado el 12 de enero de 2022, de VMWARE: https://xenproject.org

Zhang, M., Wen, H., & Hu, J. (Enero de 2016). *"Spanning Tree Protocol (STP) Application of the Inter-Chassis Communication Protocol (ICCP)", RFC 7727*. doi:10.17487/RFC7727

APÉNDICES

APÉNDICE A. Definiciones y Conceptos de las tecnologías usadas

DRBD: Distributed Replicated Block Device: en español «Dispositivos de Bloque de Replicación de Distribución». Software desarrollado para sistemas basados en GNU/Linux cuyo fin es replicar dispositivos de bloques entre dos servidores ubicados geográficamente en puntos remotos.

ESXI: Elastic Sky X Integrated: nombre de producto del hipervisor desarrollado por VMWare. Tiene la finalidad de particionar el hardware para consolidar aplicaciones, reducir costos, optimizar la administración y mejorar el rendimiento de todo el Data Center, lo cual resulta en una mayor cantidad de Servicios ofrecidos dentro de una pequeña cantidad de recursos de hardware físicos.

FTP: File Transfer Protocol: en español «Protocolo de Transferencia de Ficheros». Protocolo utilizado para la transferencia de ficheros de distintos tamaños que, a diferencia de otros protocolos, permite la reanudación de la transferencia en caso de fallos de red mediante el envío del comando REST.

HA: High Availability: en español «Alta Disponibilidad». Protocolo usado para la implementación segura de la continuidad del servicio ofrecido por un conjunto de ordenadores, en caso de fallos, se pretende que el servicio siempre ofrezca una continuidad del servicio.

HEARTBEAT: Es un servicio que se ejecuta en sistemas GNU/Linux en modo demonio, proveyendo una infraestructura de clúster, permitiendo a los clientes la desaparición o aparición de un miembro del clúster e intercambiar de forma sencilla mensajes entre los miembros.

LAN: Local Area Network: en español «Red de Área Local». Estructura de red por medio de la cual se realiza la interconexión entre dispositivos ubicados dentro de un ambiente privado y aislado del área público.

LVM: Logical Volume Manager: en español «Administrador de Volúmenes Lógicos». Gestor que es usado para optimizar la colocación de datos y acceso, abstrae el hardware mediante volúmenes lógicos, hace flexible la administración del almacenamiento de los datos, permitiendo redimensionarlos y desplazarlos sin la necesidad de detener o desmontar el sistema de archivos, administrando el espacio del disco duro con mejor eficacia que el sistema de particiones convencionales.

MDADM: Multiple Device Administrator: en español «Administrador de Múltiples Dispositivos». Software mediante el cual se administran múltiples dispositivos para la creación de un arreglo RAID, haciendo de un grupo de bloques un solo dispositivo.

NBD: Network Block Device: en español «Dispositivo de Bloque de Red». Protocolo usado para reenviar todo un dispositivo de bloques o una parte del mismo a un segundo ordenador.

NFS: Network File System: en español «Sistema de Archivos de Red». Mediante este protocolo se permite el acceso remoto de un sistema de archivos a través de la red.

OS: Operative System: en español «Sistema Operativo». Conjunto de Softwares que en compañía de un Kernel que se ejecutan a nivel privilegiado, gestionan los recursos del Hardware para darle un sentido y una funcionalidad mediante servicios ofrecidos a los programas y aplicaciones a nivel de usuario.

RAID: Redundant Array of Independent Disks: en español «Arreglo Redundante de Discos Independientes». Es una solución de almacenamiento que combina dos

o más conjunto de dispositivos de bloques y los convierte en una sola unidad lógica para proveer redundancia de datos y/o rendimiento en las operaciones de lectura y escritura de los datos.

RAVD: Redundant Array of Virtual Disks: en español «Arreglo Redundante de Discos Virtuales». Nombre designado para el proyecto desarrollado.

VM: Virtual Machine: en español «Máquina Virtual». Software que simula todo un sistema de computación permitiendo ejecutar programas de manera que aparenta estar en un hardware real.

XFS: High-performance 64-bit journaling file system: en español «Sistema de Archivos de 64 bits con Journaling de Alto Rendimiento». Sistema de archivos basado en la extensión por lo que soporta archivos muy grandes. Su límite es el espacio disponible en el sistema de archivos. Es escalable, estable, crea copias de seguridad, restaura y recupera los datos ante caídas, permite la desfragmentación y expansión estando el sistema de archivo activo y montado.

APÉNDICE B. Selección de Técnicas de Virtualización e Hipervisor

B.1. Técnicas de virtualización

De los tipos de virtualización mencionados en la sección "II.11. Virtualización" se escogió un hipervisor que implemente Virtualización Completa y que también contara con firmware o footprint[55] pequeño para que consumiera la menor cantidad de recursos posible. El motivo de haber escogido este método por sobre los demás es por las siguientes razones:

B.1.1. Abstracción de hardware

El virtualizador usa un **HAL** lo cual permite que las aplicaciones no accedan directamente al hardware físico, por el contrario, las aplicaciones lo hacen mediante una capa intermedia la cual se encarga de administrar el acceso a cualquier recurso que este siendo demandado en un tiempo determinado. Las **HAL** utilizan la misma lógica de las API con la diferencia que las **API** interactúan como interprete entre dos Software distintos, a diferencia de la interacción de las HAL son entre software (Sistema Operativo o aplicaciones) y el hardware físico.

B.1.2. Portabilidad de la máquina virtual

Gracias a la **HAL** explicada en el punto anterior, las máquinas virtuales pueden ser movidas entre distintos ordenadores físicos siempre que se respeten las restricciones de compatibilidad arquitectural, por ejemplo: En una arquitectura de máquina x86 (32 bits) únicamente se pueden virtualizar máquinas con sistemas operativos x86, por el contrario, en una arquitectura de x64 (64 bits) se pueden

[55] **FootPrint:** es el espacio que una unidad de hardware o software ocupa.

virtualizar máquinas con sistemas operativos x86 y x64 debido a que la arquitectura lo permite.

Por lo explicado anteriormente es importante conocer la arquitectura de máquina donde se encuentra alojado el sistema anfitrión y la arquitectura de la máquina virtual a migrar.

B.2. Hipervisor

Para seleccionar el hipervisor era necesario que permitiera Virtualizar Completamente un ordenador, los hipervisores que permiten esta técnica de virtualización son los siguientes:

- Citrix XenServer
- Linux KVM
- VMWare ESXI
- Microsoft Hyper-V Server
- Oracle VM Server

De estos virtualizadores seleccionamos VMWare ESXI, principalmente por que la empresa REKASA usa este tipo de hipervisor, otro motivo es que este hipervisor ofrece estabilidad y seguridad gracias a su footprint que es cerca de 32 MB creado única y exclusivamente con la finalidad de servir como sistema anfitrión para la virtualización, esto quiere decir que no se es necesario protección adicional como un **OS** de uso genérico, evitando de esta manera la existencia de fallos o bugs residentes en el sistema operativo anfitrión.

APÉNDICE C. Selección del Sistema Operativo

Existen varias distribuciones GNU/Linux, entre estas Red Hat, Centos, Ubuntu, Fedora, Debian, OpenSUSE, etc. Para escoger la distribución podemos tomar de ejemplo las más populares mostradas en el siguiente gráfico:

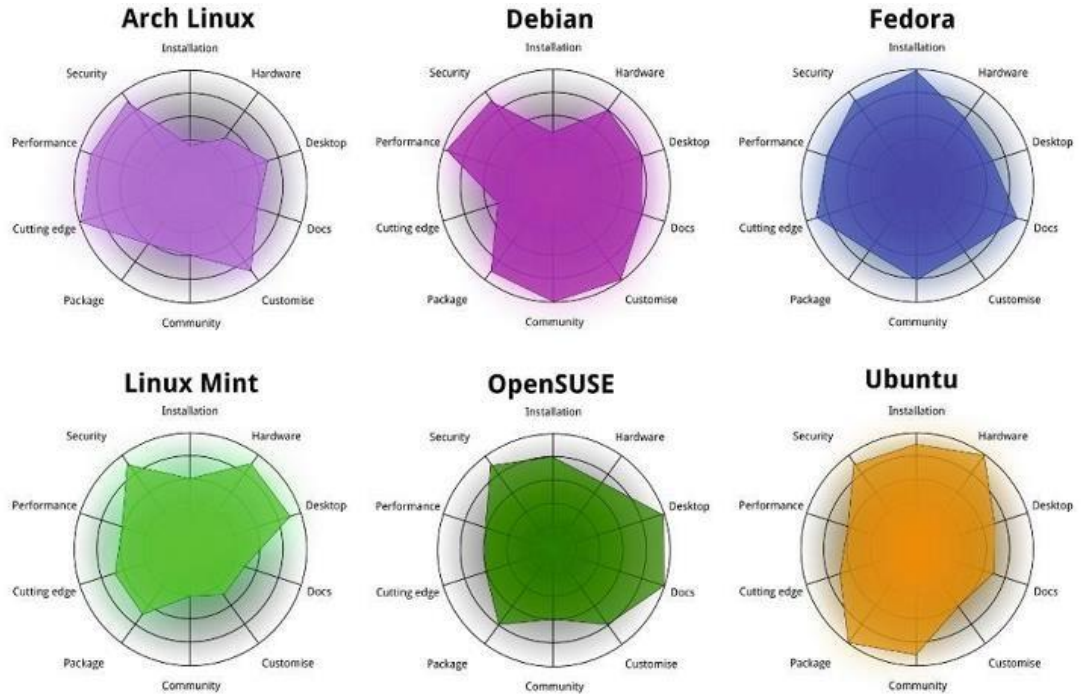


Figura (40) - Comparación entre distribuciones más populares

Se escogió Debian como sistema operativo a utilizar de entre las distintas distribuciones ofrecidas para GNU/Linux, porque Debian es una organización y no una compañía, no vende nada, sus miembros son voluntarios, todos los desarrolladores trabajan bajo una misma meta: **"El mejor sistema operativo de todos"**, de esta forma hemos tomado en cuenta cada uno de los siguientes criterios:

C.1. Filosofía de la distribución

Cada proyecto o distribución tiene una filosofía propia o motivo por el cual ha nacido y cuya metas y objetivos se ven sujetas a esta filosofía, así por ejemplo la distribución de Ubuntu tiene la siguiente filosofía: **"Ubuntu: Linux for human beings"**, esto conlleva a que el sistema operativo Ubuntu siempre tenga un diseño amigable para el usuario y todas sus distribuciones van pensadas en esto, por otro lado la filosofía de Debian es **"The universal operating system"**, donde están comprometidos a usar mayoritariamente software completamente libre y cuya meta es funcionar de forma estable en casi todos los ordenadores y arquitecturas.

Así puede suceder que cada proyecto puede cambiar fácilmente con el tiempo su metodología de uso, su confiabilidad y su disponibilidad, etc. Pero su filosofía siempre es la misma.

Algunos podrían decir que usan Debian porque es gratuito, porque parece genial o sin costo, y estas percepciones son válidas, pero se pierde el sentido de lo que es en verdad el software libre y su máxima razón de ser. La razón del software libre es el resultado evolutivo de la investigación académica que se sostiene y crece gracias a la libertad del software. Si no existiese la libertad del software, significaría que cada vez que se requiera dar solución a un determinado problema se tendría que reescribir el código desde un inicio en lugar de tomar un código ya existente y ajustarlo en beneficio de dicho problema. Por lo tanto, no se puede ni se debe minorar a escoger una distribución por ser más gratis que otra (es decir con menor

costo), ya que lo que las diferencian no son estas cualidades superficiales, sino las basados en su filosofía que es en sí lo que la distribución ofrece.

Debido a esto las filosofías de cada distribución mantienen y ofrecen algo más que un software excelente y gratuito, ofrecen un objetivo para satisfacer una necesidad.

C.1.1. Soporte

Debian posee una amplia gama de desarrolladores, así como también una gran comunidad que respalda al usuario.

C.1.2. Estabilidad

Debian ofrece una excelente estabilidad, la cual se ha logrado comprobar al brindar los servicios ofrecidos por la empresa REKASA, la cual tiene más de 15 años trabajando con esta distribución, tiempo en el cual solo se ha requerido mantener actualizando la versión de la distribución y su paquetería.

Una de las características estables de esta distribución es que no desaparecerá simplemente por su gran tamaño, como lo han hecho muchas otras distribuciones, por lo que no podría uno quedarse sin soporte, tampoco puede caer en bancarrota ya que su contrato social le prohíbe abruptamente decidir a dejar de dar soporte a herramientas cuyas versiones son no empresariales de la distribución. Así nuestro sistema operativo siempre será estable y no caerá rehén de planes de negocios y empresas.

Debian ofrece tres grados de liberación para su proyecto y desarrollo de los cuales el usuario puede libremente escoger cada grado a voluntad propia:

- ✓ **Unstable:** es para prueba y desarrollo, no se ofrece al público, los paquetes de esta son luego propagados a la rama Testing.

- ✓ **Testing:** es generada automáticamente mediante scripts que intentan pasar de una a otra los paquetes que son considerables con pocos niveles de fallos provenientes de la rama Unstable, siempre y cuando las dependencias pueden ser satisfechas siempre.
- ✓ **Stable:** es la versión oficial y es usada por los usuarios ordinarios.

C.1.3. Calidad de implementación

Debian ofrece herramientas que una vez probadas atrae a la mayoría de los usuarios como por ejemplo su herramienta "apt-get" que no solo hace que la experiencia sea genial, sino que también sea sumamente fácil instalar un software en un sistema operativo Debian.

Este es el punto integral y clave, el corazón palpitante de Debian y lo que lo hace tan absolutamente superior a todos los otros sistemas operativos. La política se define y es bien clara. Se aplica a través de las herramientas que utiliza todos los días. Cuando se ejecuta "apt-get install foo", no solo se está realizando la instalación de un software, sino que se encarga que todo funcione correctamente cumpliendo todas las dependencias y requerimientos que deban ser instalados, así no solo se realiza una simple instalación, también se está satisfaciendo una política que cumpla con el objetivo de dar el mejor sistema posible.

Lo que define la política son los límites de Debian, y no las acciones del usuario en el sistema. La Política define qué partes del sistema, el gestor de paquetes puede cambiar, cómo manipular los archivos de configuración, etc. Al limitar el alcance de la distribución de esta manera, el administrador del sistema puede realizar modificaciones sin temor a que los cambios realizados afecten archivos fuera de la zona establecida en que los paquetes de Debian deban realizar sus cambios. En esencia, cuando los paquetes violan su política definida, se introduce del lado del desarrollo una nueva clase de errores de política, estos pasan a ser presentados con severidad crítica, por consiguiente, el paquete afectado no será

incluido en la versión estable oficial de Debian, quedando solo en las versiones testing y unstable.

En comparación con las distribuciones comerciales de Linux, Debian tiene muchos mayores desarrolladores disponibles para empaquetar proporciones grandes de paquetes. Sumado a que Debian carece de ciclos donde los plazos sean impulsados con fines de negocios, tendiendo a hacer las cosas bien y no para producir al mercado distribuciones estables con plazos establecidos con afinidad meramente comerciales.

C.1.4. Repositorios

Debian posee actualmente más de 29,000 paquetes. Las probabilidades de que todo lo que necesita el usuario ya esté empaquetado para debían son muy altas, teniendo así al alcance un vasto número de paquetes disponibles para el usuario.

C.1.5. Documentación

Debian goza de una vasta paquetería acompañada de una amplia documentación, ambas son alimentadas por una gran comunidad de desarrolladores.

C.1.6. Mantenimiento y administración

La actualización en Debian es la envida de la mayoría del resto de distribuciones, se puede tener un sistema operativo Debian en un ordenador que durante 5 años no haya tenido que ser reinstalado en cada actualización.

La administración es tan simple que basta con hacer un “apt-get install sendmail” y tener completamente un servicio con **SASL**, **TLS**, configurado completamente con certificados completos. Se garantiza la preservación de la configuración ante cualquier actualización, dejando centralizado todos los archivos de configuración en el directorio /etc/, permitiendo así una forma fácil para respaldar los datos.

C.1.7. Portabilidad y soporte de Hardware

Debian al usar el kernel **GNU/Linux**, soporta una mayor amplia gama de hardware esotérico que **BSD**, así por ejemplo algo ya soportado por **GNU/Linux** como hardware **RAID** que son las tarjetas **3ware RAID**[56], apenas empiezan a ser soportadas por **BSD** recientemente. **IBM** y **HP** aseguran que su hardware está completamente soportado por **GNU/Linux**. Para los ordenadores usados para escritorios, el único tema complejo son los drivers exóticos y casi únicos, pero fuera de eso, no existen mayores inconvenientes.

C.1.8. Seguridad y confiabilidad

Siempre ha existido una relación entre seguridad y comodidad. La computadora más segura del mundo es aquella que nunca ha sido encendida. Segura, pero no tan útil.

¿Qué piensa uno cuando se habla de seguridad en Sistemas Operativos al estilo **UNIX**? Se nos viene a la mente **OpenBSD**, sin embargo, a pesar de la seguridad tendremos software muy obsoleto. Muchos están de acuerdo que la porción auditable y segura de **OpenBSD** no provee el software que requieren.

Existe un foco muy enorme en la seguridad y estabilidad en GNU/Linux Debian, y en la distribución estable se reclaman las mismas particularidades de la reputación de OpenBSD, y existen aparentemente algunas diferencias en el mundo real entre Debian y OpenBSD en este momento. Basta con trabajar un poco y pasar unos breves minutos endureciendo los paquetes estándares y las prioridades de solo usar paquetes útiles y necesarios.

---

[56] **3ware RAID CARDS:** es una marca de hardware para crear y configurar discos RAID. Ver más detalles: http://www.avagotech.com/products/server-storage/raid-controllers/

Aunque no sea necesaria una cadena de comandos en cada sistema BSD destinado para desplegar actualizaciones de seguridad ("make release", o "make package" para construir en una maquina e instalar en cualquier otra parte), la forma en que BSD maneja el despliegue de sus paquetes es un poco más complejo comparado con Debian. La distribución de paquetes binarios es más eficiente en Debian, debido a que uno puede tener su propio archivo apto para apt[57] y utilizarlo en todos los servidores productivos, usando los mecanismos nativos de Debian.

Sin embargo, cuando se trata de seguridad real, sin controles de acceso obligatorios no hay mucha garantía. SELinux sería una mejor opción que OpenBSD o que Debian basado solo con paquetes estables.

Aún sin SELinux, se puede encontrar la estabilidad de Debian sólida como una roca, con la tranquilidad de saber que vienen de parches de seguridad que han sido portados a la distribución de parte del equipo de seguridad.

Existe otro beneficio que nos ofrece el equipo de seguridad de Debian cuando nos referimos a la distribución estable, el cual es la posibilidad de tener múltiples versiones para cada suite, por ejemplo: Apache, KDE y X11, se puede tener una versión distinta para cada suite con actualizaciones de seguridad para la distribución estable, esto no es posible en OpenBSD, sin embargo debemos recordar que solo existe una versión del árbol de puertos, por lo que cada versión de la misma suite debe de funcionar en un puerto único, es decir si el puerto 80 está ocupado por apache2.2, al tener otra versión para la misma suite por ejemplo apache2.4 se le debe asignar otro puerto distinto del 80 como el 8080 para esta otra versión, esto tampoco es posible en OpenBSD.

---

[57] **apt:** Programa de gestión de paquetes creado por el proyecto Debian donde la abreviatura “apt” significa Advanced Packaging Tool. Este programa simplifica la instalación y eliminación de paquetes en la distribución.

OpenBSD debe instalar la nueva versión del paquete con todos los posibles problemas que esto pueda causar, como lo pueden ser actualizaciones a todas las dependencias, así como horas en el cambio de líneas de código, a diferencia de Debian que se puede instalar la misma versión del paquete, con las revisiones de seguridad o parches de corrección para no tener que instalar una versión más actual.

Podemos encontrar una comparación entre las distribuciones de Linux y el tiempo para parchar las vulnerabilidades de seguridad conocidas, pero no están incluidas los datos de BSD, por las razones previamente dadas, sin embargo, aquí está la URL:

https://people.debian.org/~jfs/debconf3/security/data/

C.1.9. Escalabilidad y desempeño

El desempeño varía entre una distribución liberada a otra. Podemos encontrar un conjunto completo de pruebas benchmark, con su código, el cual está disponible en la siguiente URL:

http://bulk.fefe.de/scalability/

En dicha URL se puede observar que Linux 2.6 escala en todas las pruebas.

C.1.10. Conclusión de nuestra elección

Después de verificar todos los puntos anteriores y analizar las cualidades que nos ofrece Debian es muy difícil que otra distribución nos ofrezca la facilidad de mantenimiento, administración, accesibilidad, estabilidad, tamaño, soporte,

documentación, portabilidad de hardware, seguridad, confiabilidad, escalabilidad, desempeño, capacidad de personalización y un fuerte apoyo.

Debian nos permite enfocar al usuario en lograr tener hecho su trabajo, fácil, seguro y con la mínima preocupación sobre la infraestructura usada, lo cual Debian ayuda a cumplir con esto.

Hasta hoy Debian permite actualizar, instalar y manejar paquetes de forma sencilla y estable, mejor que cualquier otra distribución existente.

El sistema operativo Debian Jessie se adquirió de la siguiente url:

https://cdimage.debian.org/cdimage/archive/8.11.1/amd64/iso-dvd/

APÉNDICE D. Creación de Máquinas Virtuales en Workstation

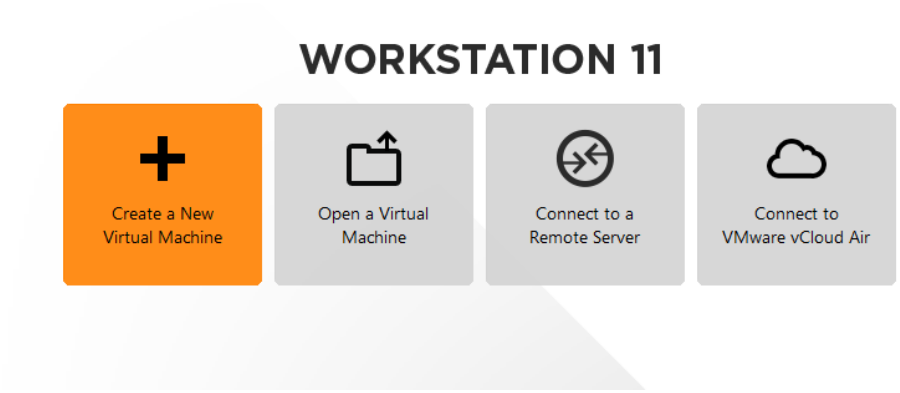


Figura (41) - Creación máquina virtual Workstation (Screenshoot 01)

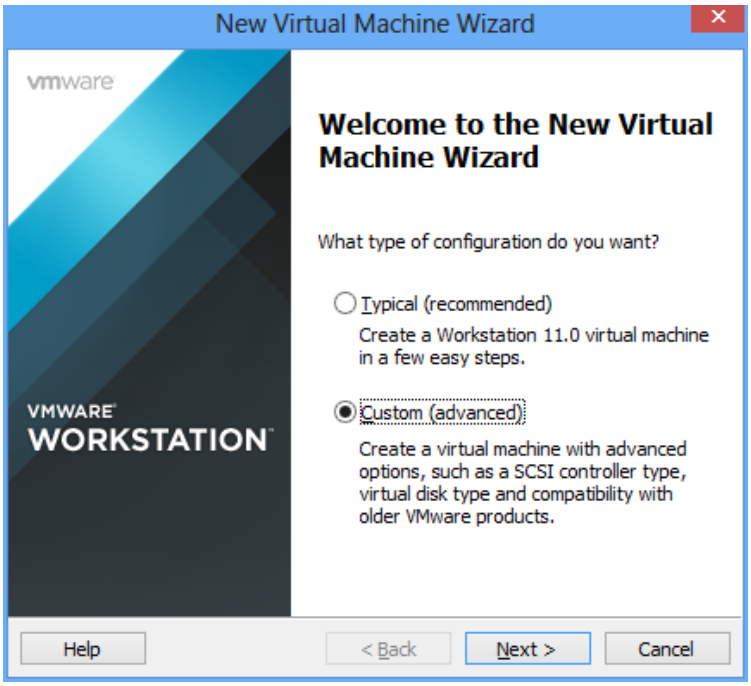


Figura (42) - Creación máquina virtual Workstation (Screenshoot 02)

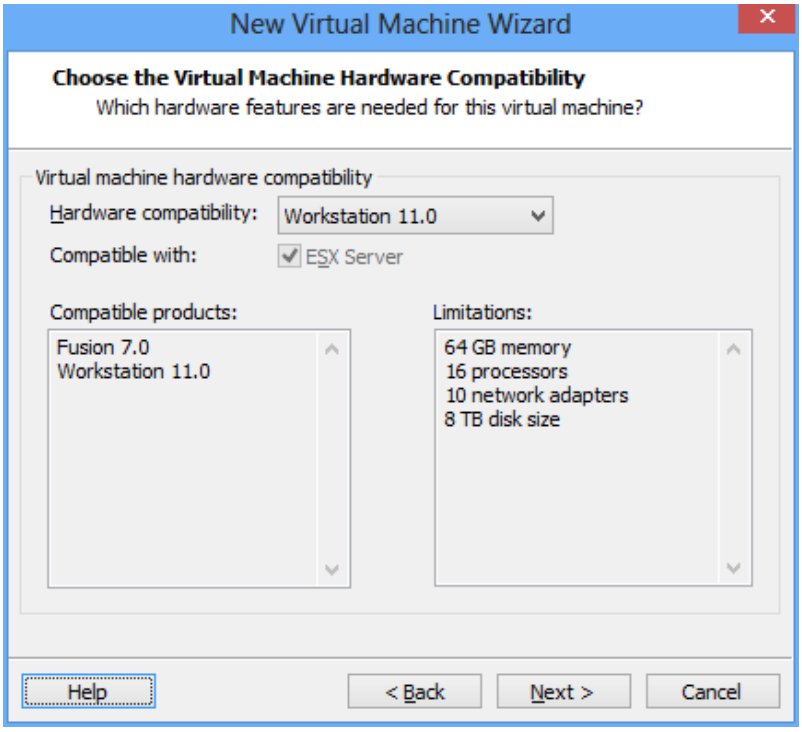


Figura (43) - Creación máquina virtual Workstation (Screenshoot 03)

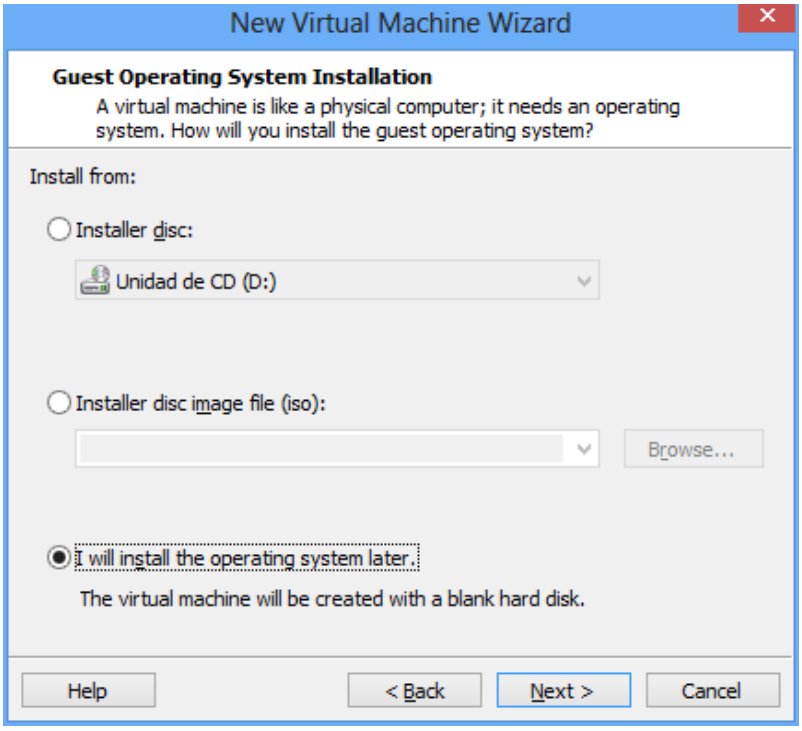


Figura (44) - Creación máquina virtual Workstation (Screenshoot 04)

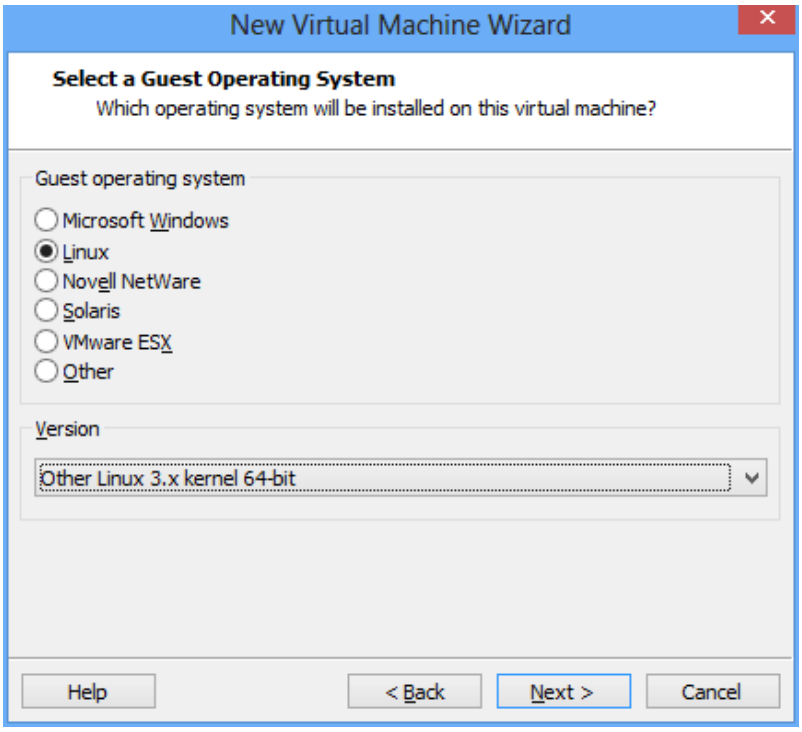


Figura (45) - Creación máquina virtual Workstation (Screenshoot 05)

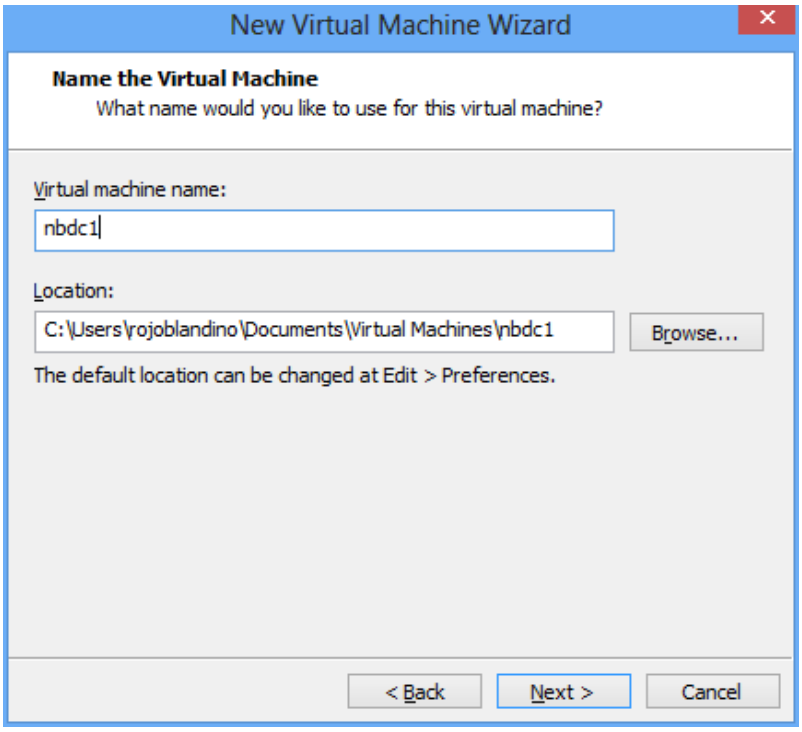


Figura (46) - Creación máquina virtual Workstation (Screenshoot 06)

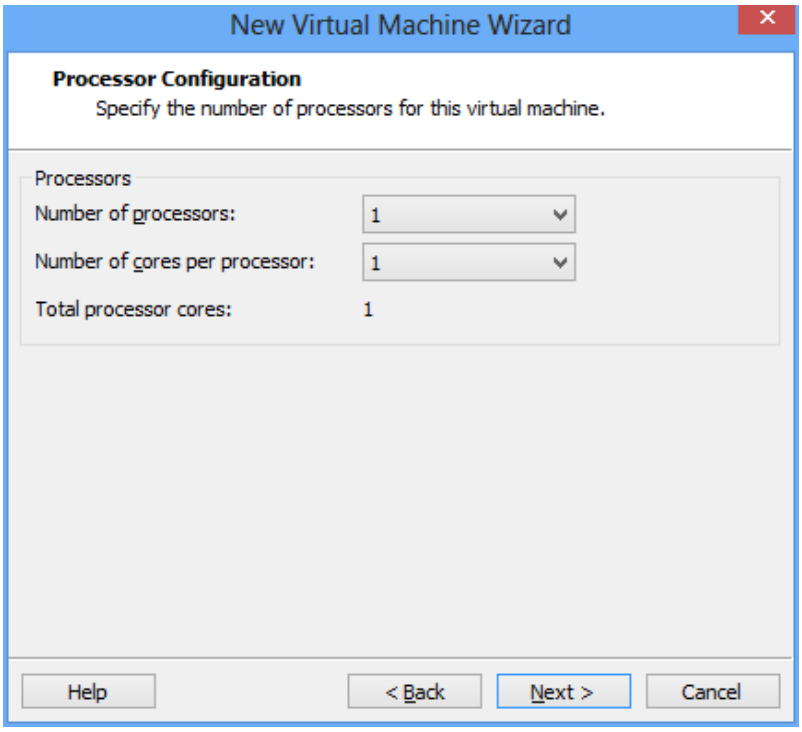


Figura (47) - Creación máquina virtual Workstation (Screenshoot 07)

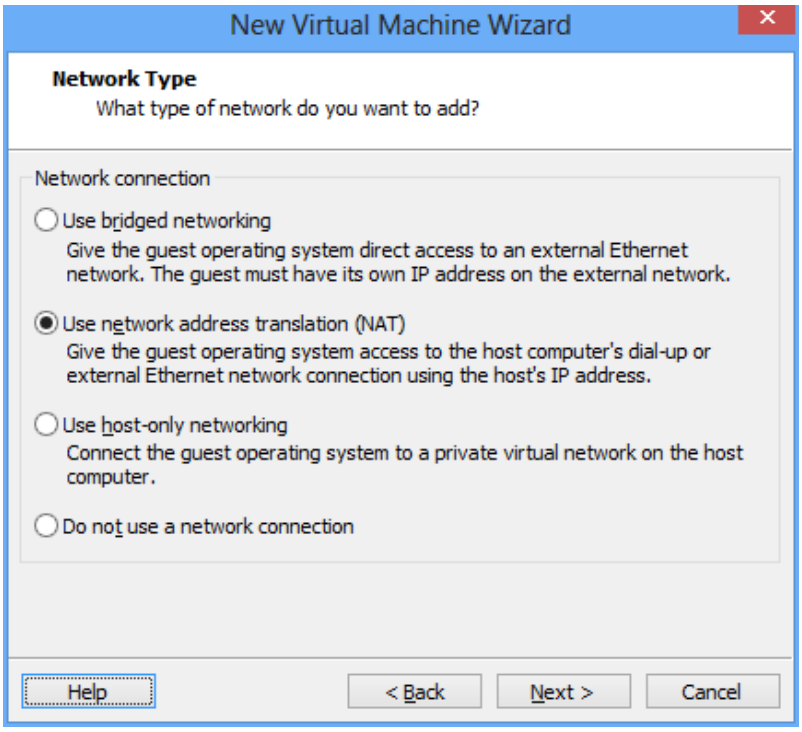


Figura (48) - Creación máquina virtual Workstation (Screenshoot 08)

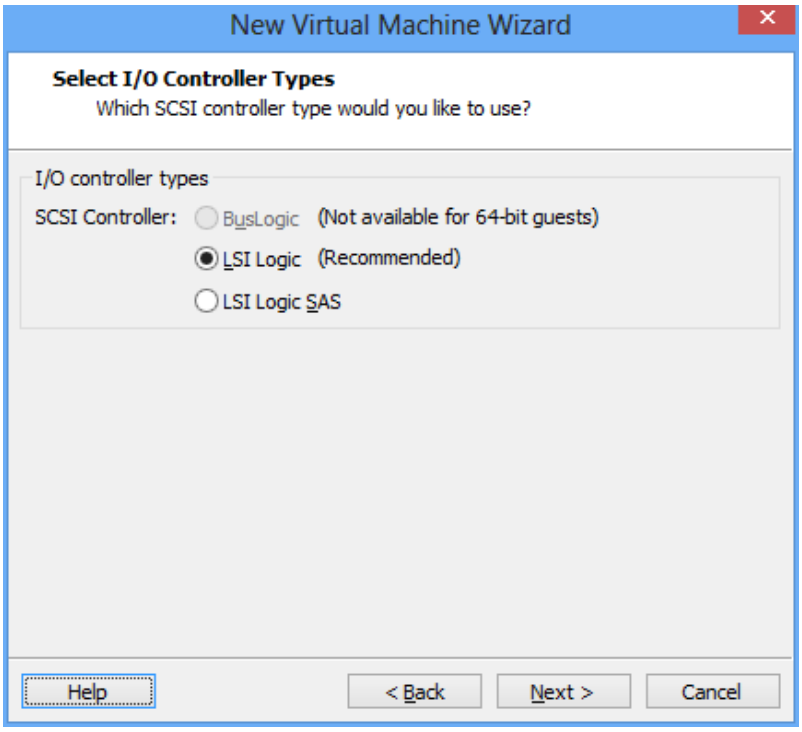


Figura (49) - Creación máquina virtual Workstation (Screenshoot 09)

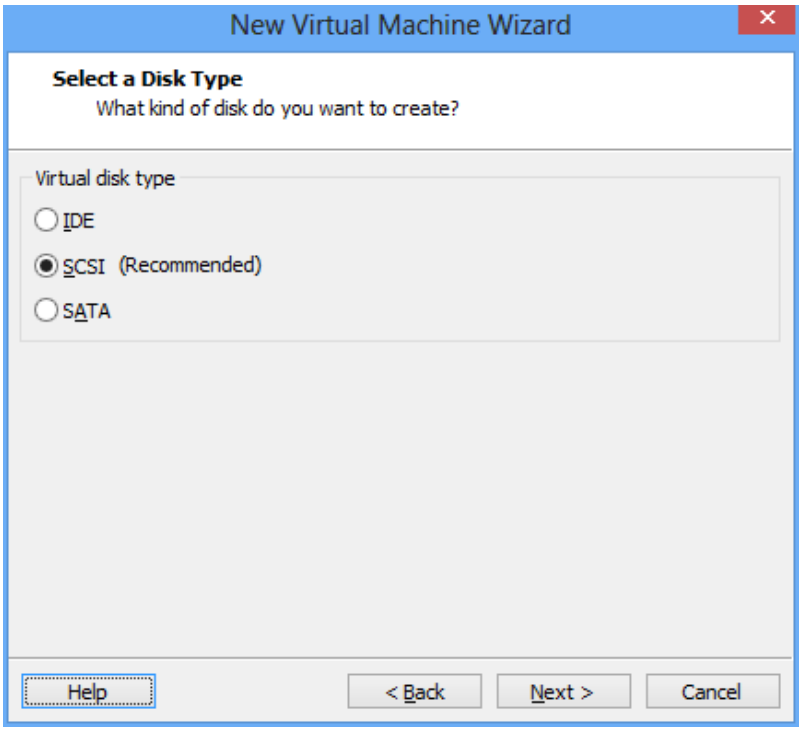


Figura (50) - Creación máquina virtual Workstation (Screenshoot 10)

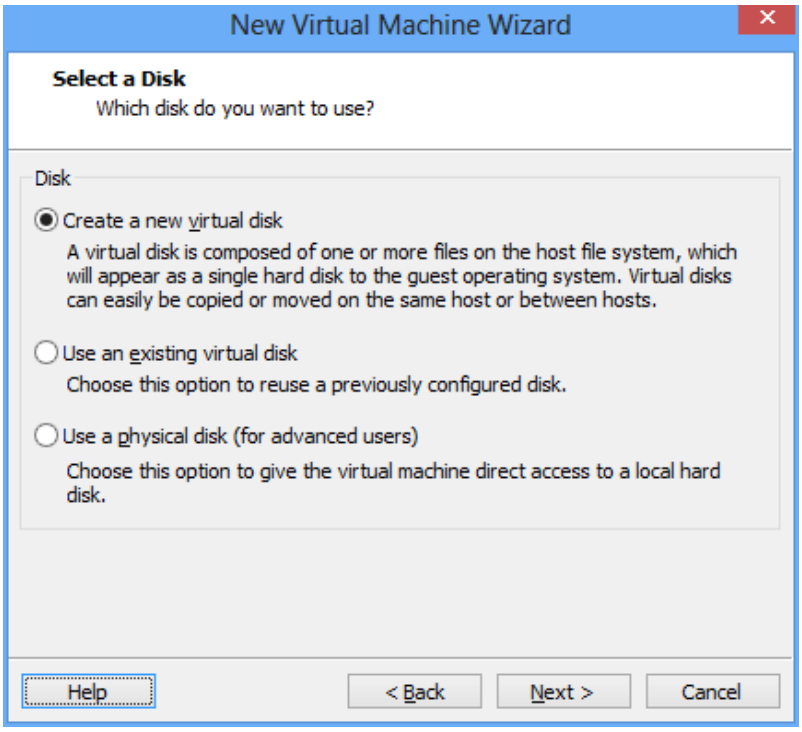


Figura (51) - Creación máquina virtual Workstation (Screenshoot 11)

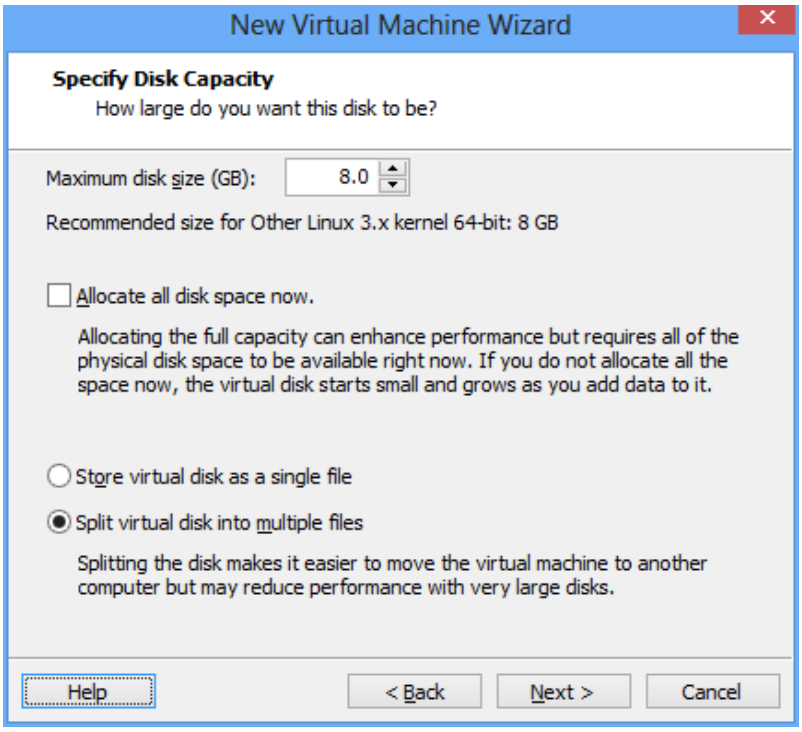


Figura (52) - Creación máquina virtual Workstation (Screenshoot 12)

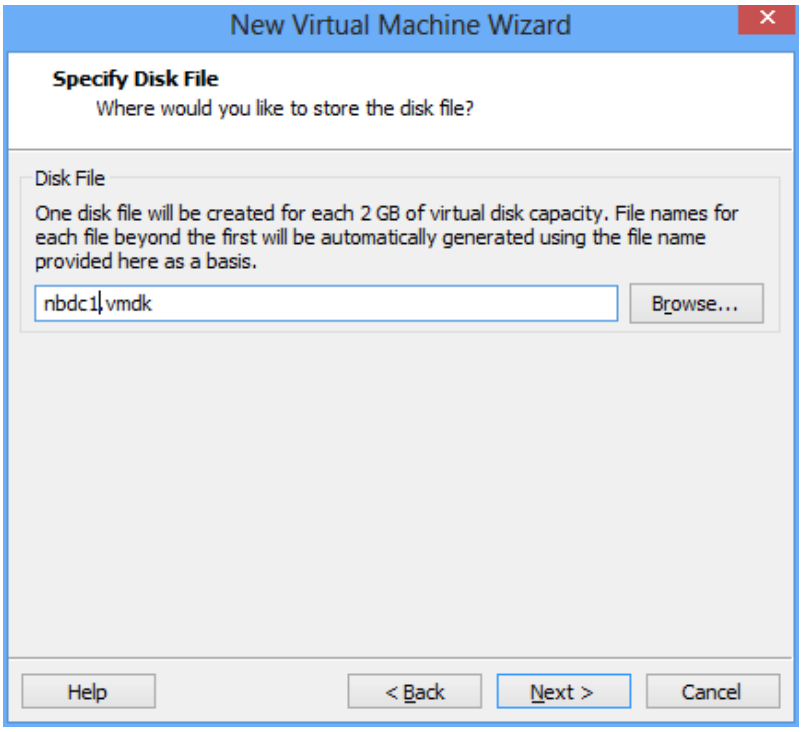


Figura (53) - Creación máquina virtual Workstation (Screenshoot 13)

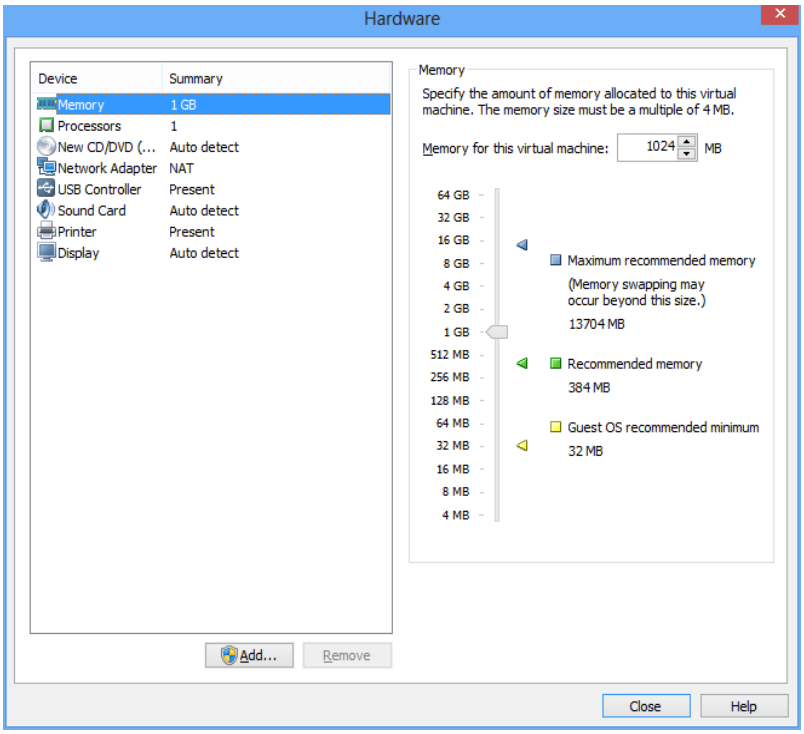


Figura (54) - Creación máquina virtual Workstation (Screenshoot 14)

APÉNDICE E. Creación de Máquinas Virtuales en ESXI

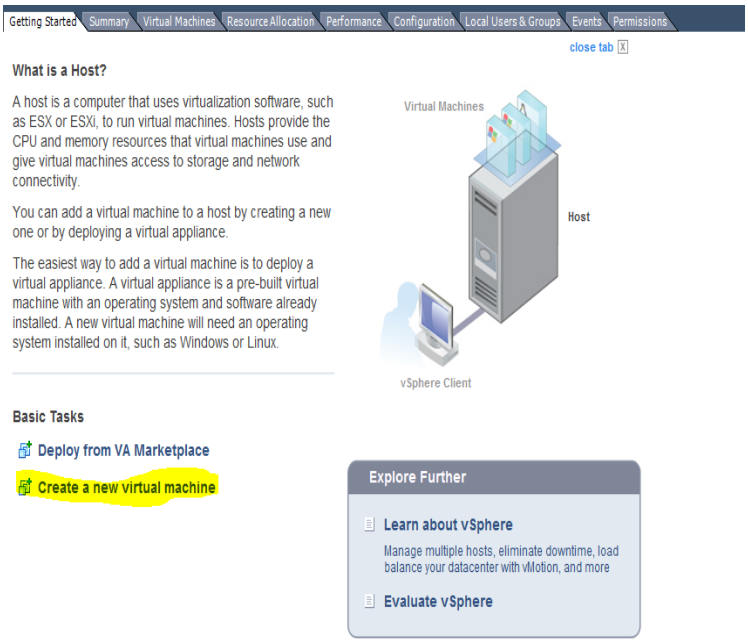


Figura (55) - Creación máquina virtual ESXI (Screenshoot 01)

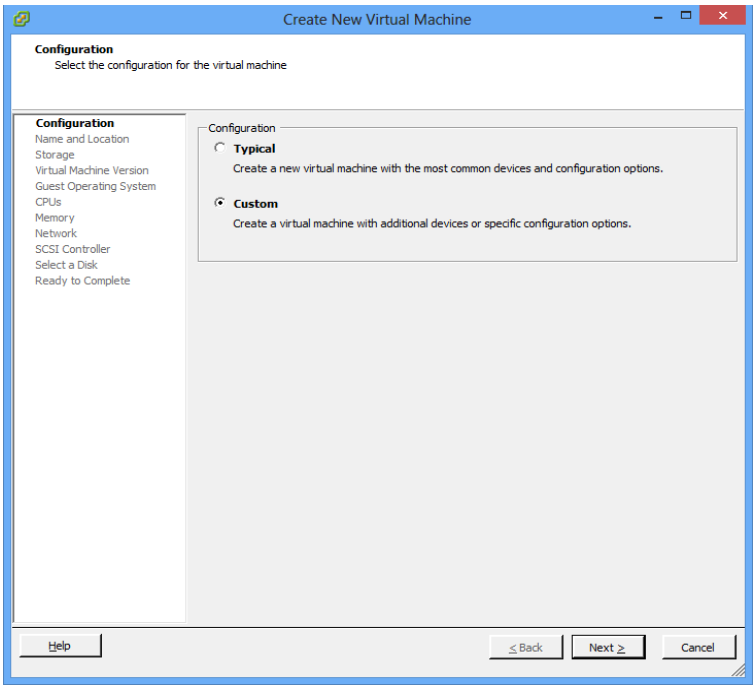


Figura (56) - Creación máquina virtual ESXI (Screenshoot 02)

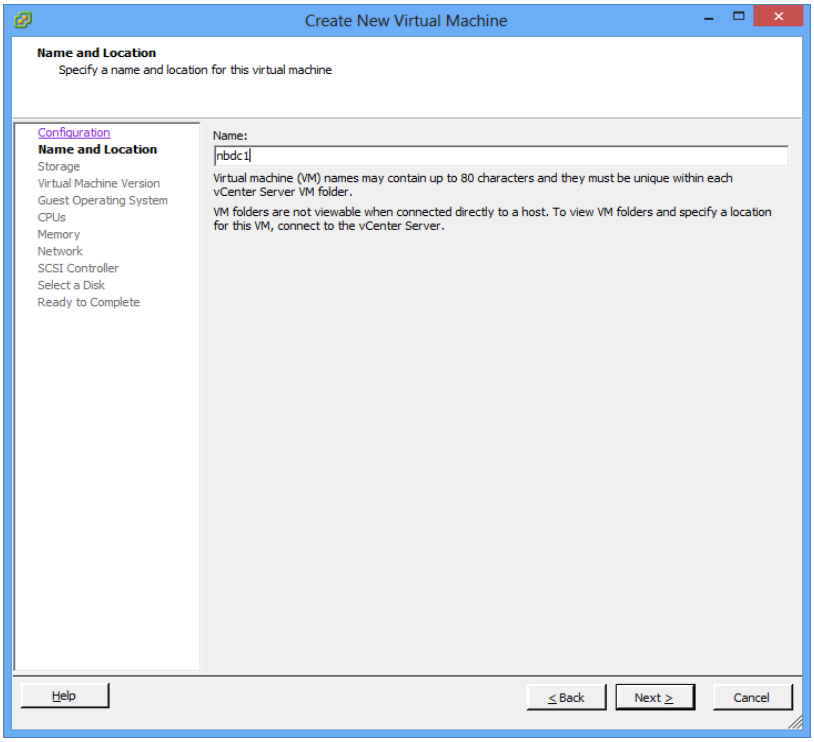


Figura (57) - Creación máquina virtual ESXI (Screenshoot 03)

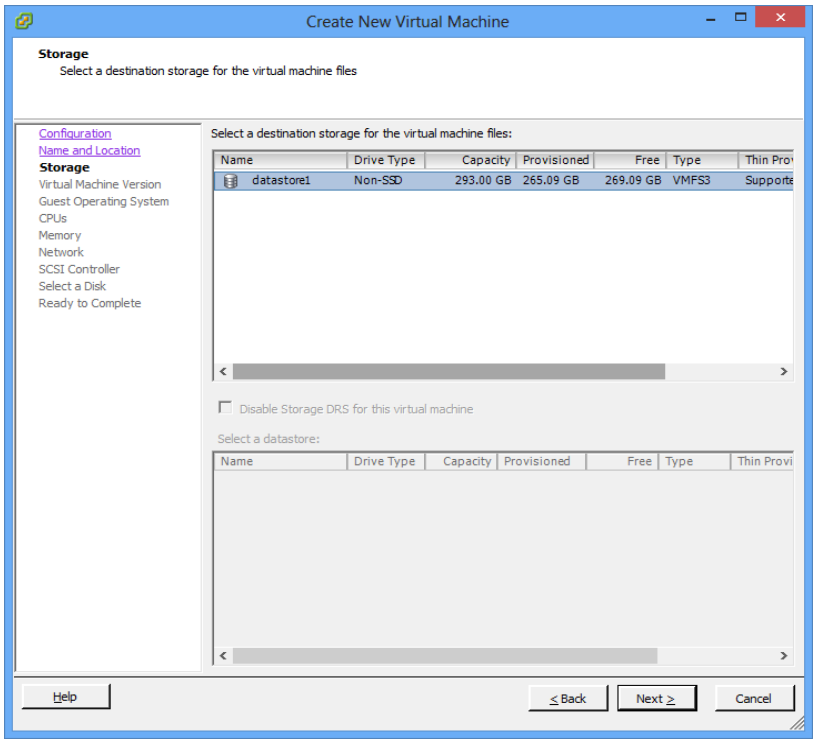


Figura (58) - Creación máquina virtual ESXI (Screenshoot 04)

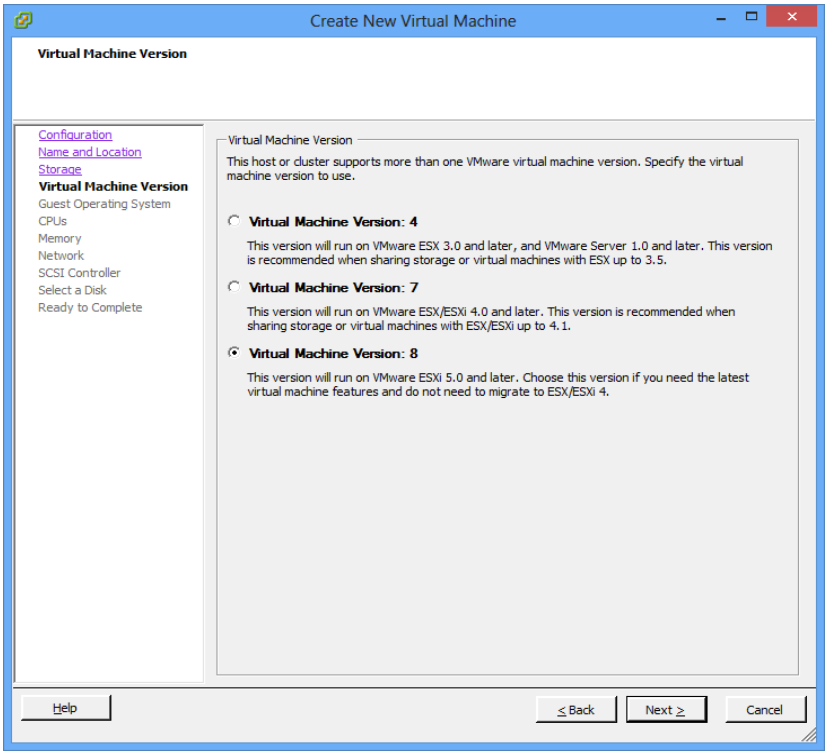


Figura (59) - Creación máquina virtual ESXI (Screenshoot 05)

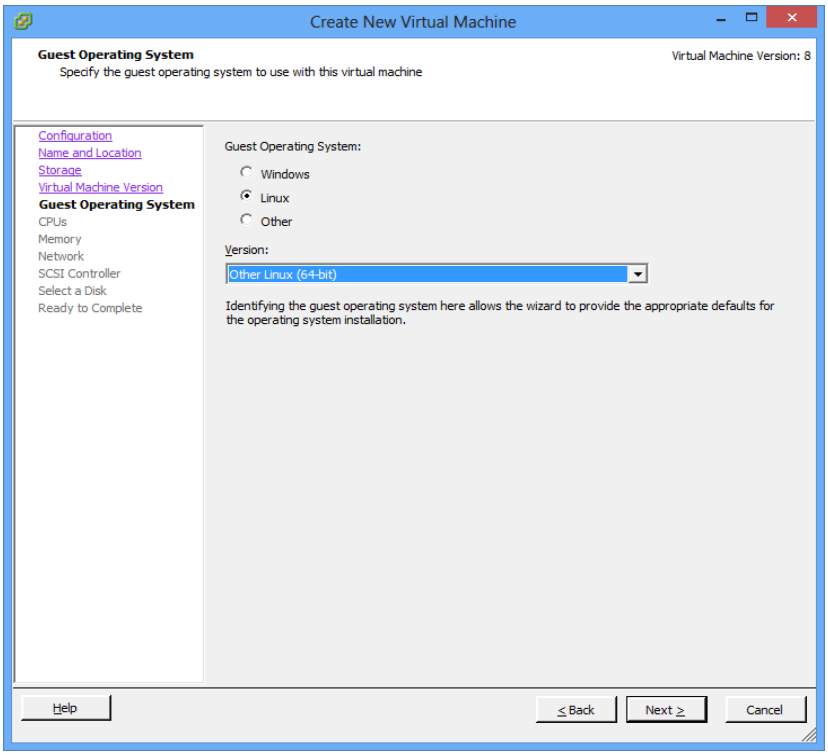


Figura (60) - Creación máquina virtual ESXI (Screenshoot 06)

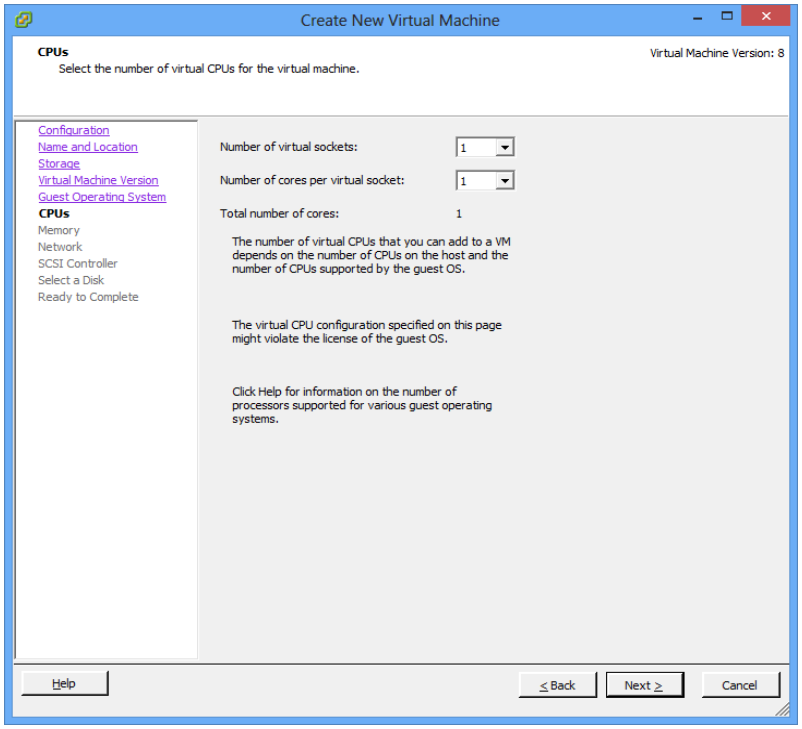


Figura (61) - Creación máquina virtual ESXI (Screenshoot 07)

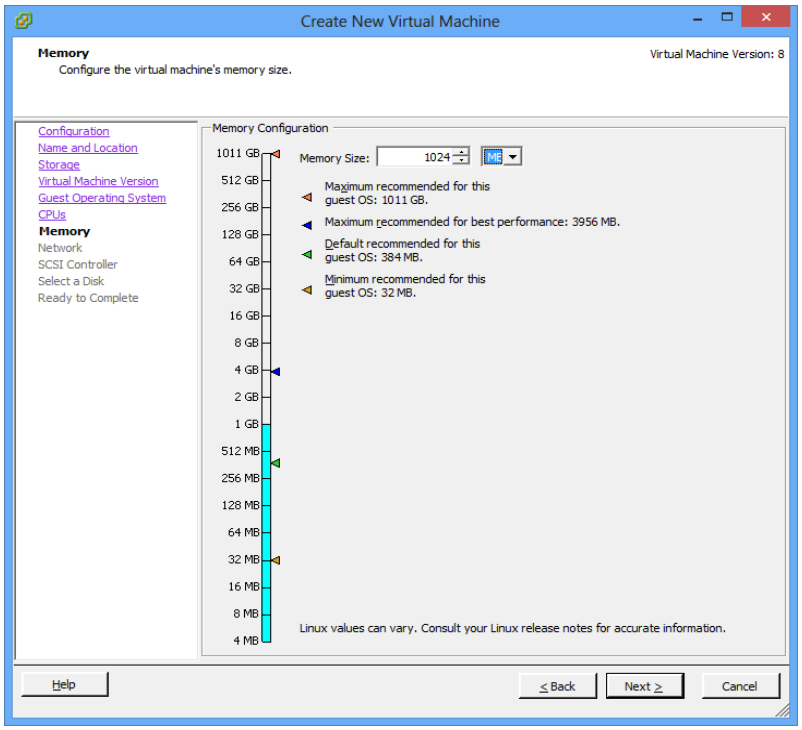


Figura (62) - Creación máquina virtual ESXI (Screenshoot 08)

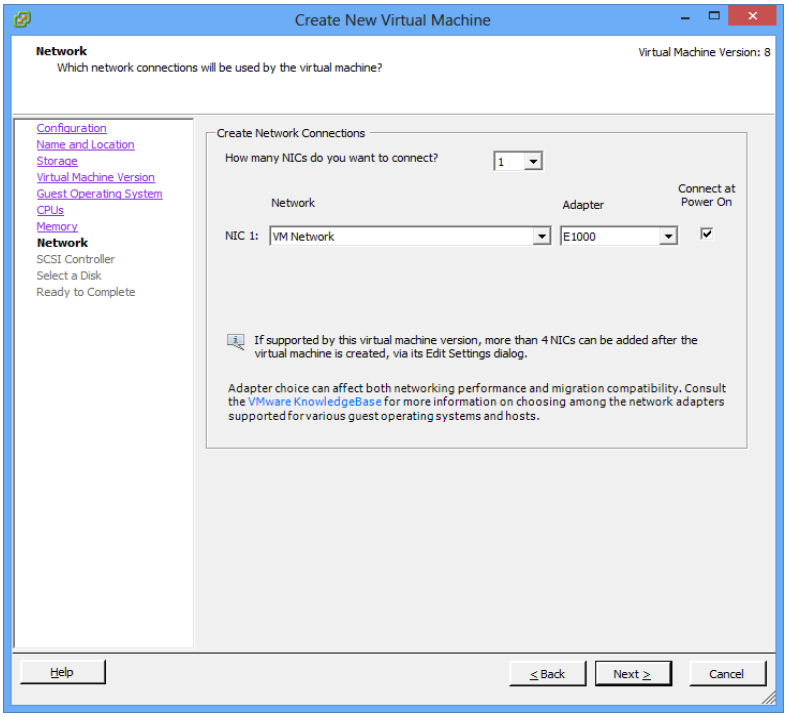


Figura (63) - Creación máquina virtual ESXI (Screenshoot 09)

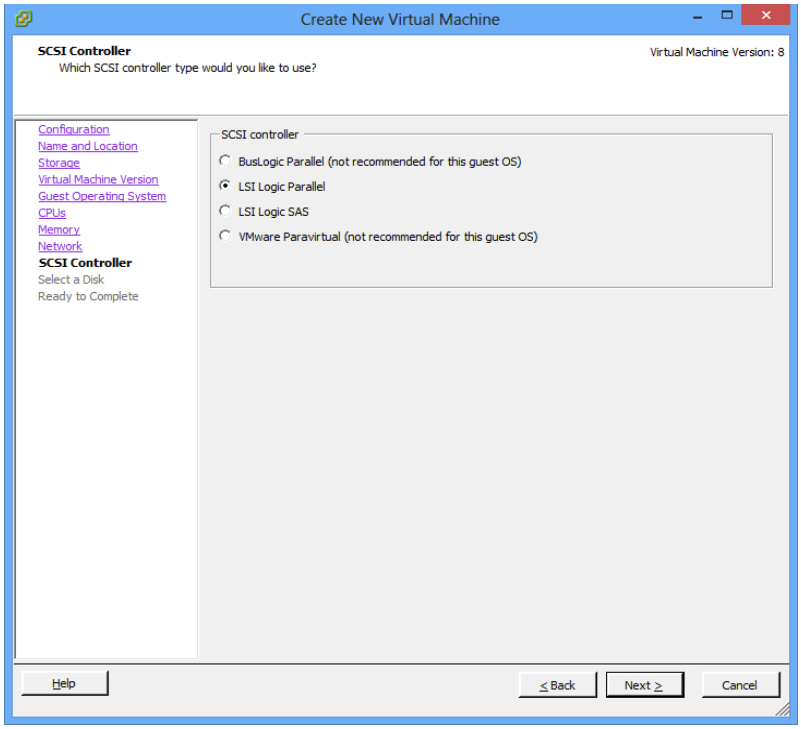


Figura (64) - Creación máquina virtual ESXI (Screenshoot 10)

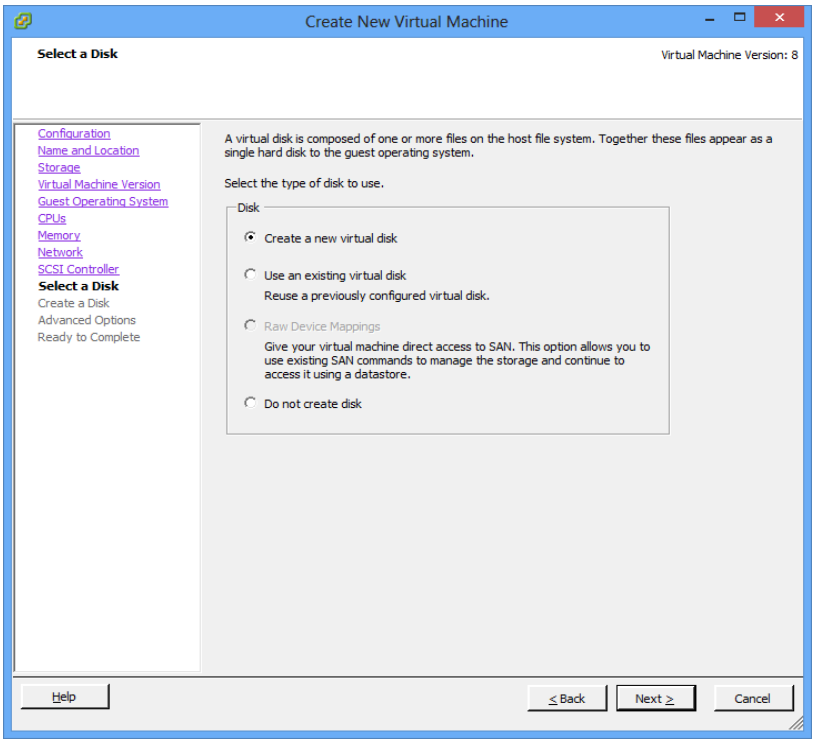


Figura (65) - Creación máquina virtual ESXI (Screenshoot 11)

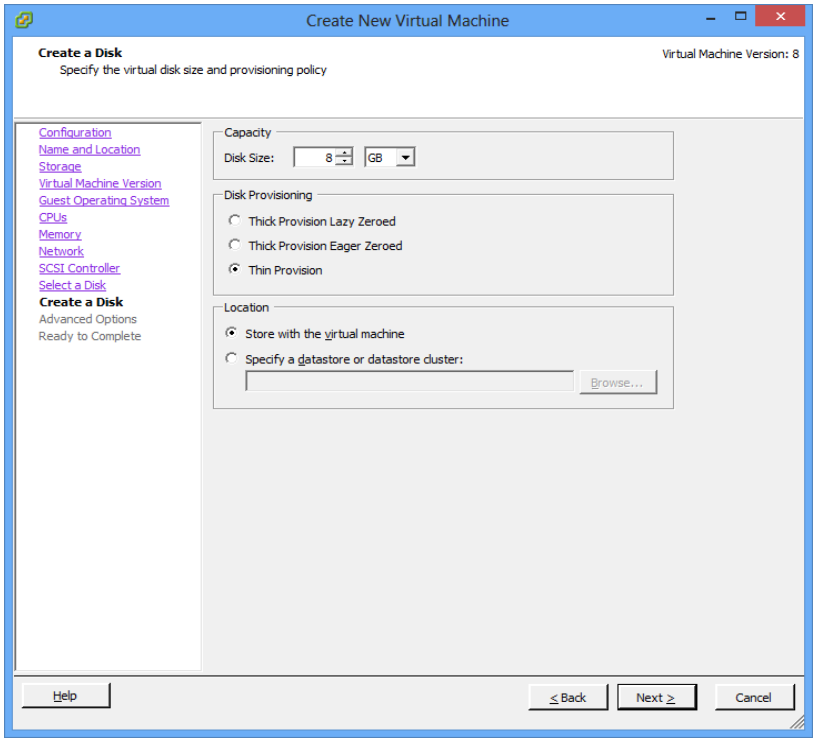


Figura (66) - Creación máquina virtual ESXI (Screenshoot 12)

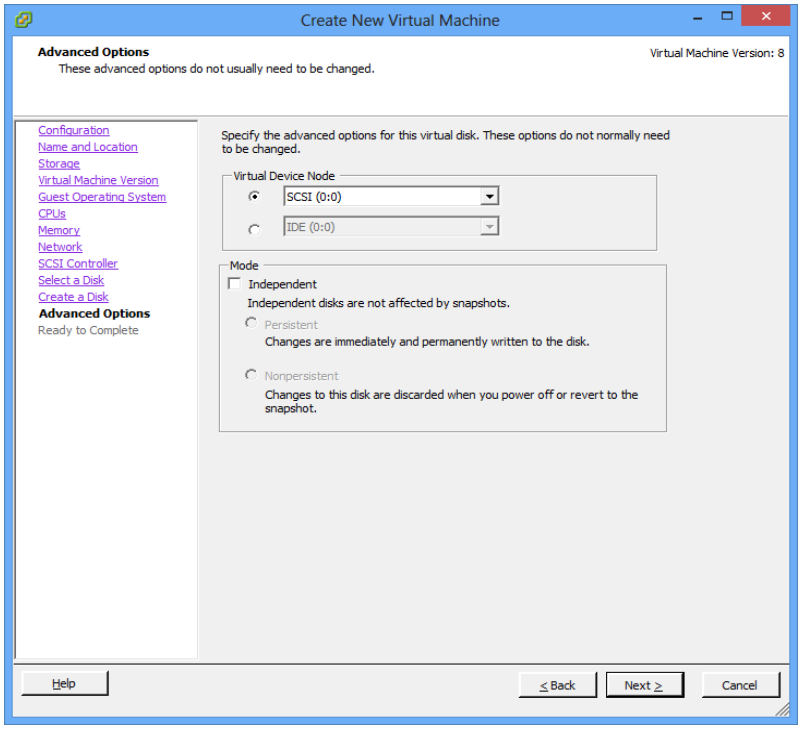


Figura (67) - Creación máquina virtual ESXI (Screenshoot 13)

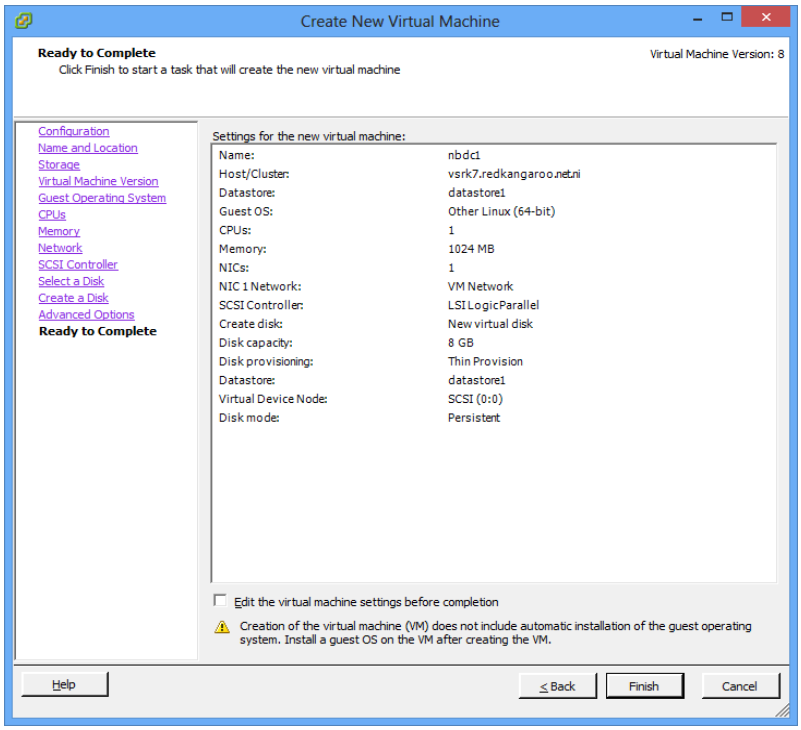


Figura (68) - Creación máquina virtual ESXI (Screenshoot 14)

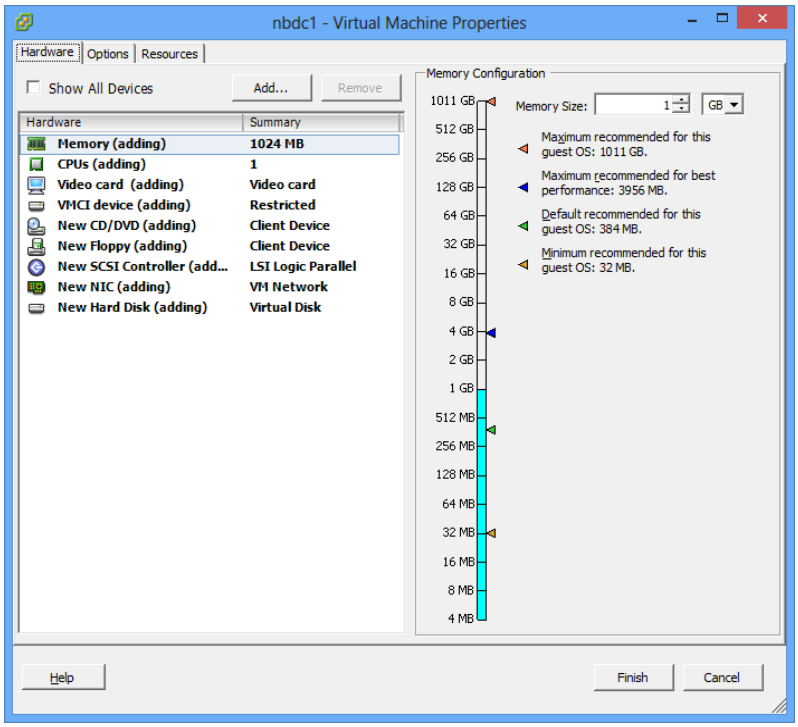


Figura (69) - Creación máquina virtual ESXI (Screenshoot 15)

APÉNDICE F. Configuraciones de Red

Cuadro (115) - Configuración de red para nbdc1

Path: /etc/network/interfaces

```
...
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 172.16.2.10
netmask 255.255.255.0
```

Cuadro (116) - Configuración de red para nbdc2

Path: /etc/network/interfaces

```
...
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 172.16.2.11
netmask 255.255.255.0
```

Cuadro (117) - Configuración de red para nbds1

Path: /etc/network/interfaces

```
...
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 172.16.2.20
netmask 255.255.255.0
```

Cuadro (118) - Configuración de red para nbds2

Path: /etc/network/interfaces

```
...
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 172.16.2.21
netmask 255.255.255.0
```

Cuadro (119) - Configuración de red para nbds3

Path: /etc/network/interfaces

```
...
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 172.16.2.22
netmask 255.255.255.0
```

Cuadro (120) - Configuración de red para nbds4

Path: /etc/network/interfaces

```
...
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 172.16.2.23
netmask 255.255.255.0
```

Cuadro (121) - Configuración de red para nbds5

Path: /etc/network/interfaces

```
...
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 172.16.2.24
netmask 255.255.255.0
```

Cuadro (122) - Configuración de red para nbds6

Path: /etc/network/interfaces

```
...
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 172.16.2.25
netmask 255.255.255.0
```

Cuadro (123) - Configuración de red para nbds7

Path: /etc/network/interfaces

```
...
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 172.16.2.26
netmask 255.255.255.0
```

Cuadro (124) - Configuración de red para nbds8

Path: /etc/network/interfaces

```
...
allow-hotplug eth0
iface eth0 inet dhcp

auto eth1
iface eth1 inet static
address 172.16.2.27
netmask 255.255.255.0
```

Cuadro (125) - Configuración de red para nbds9

Path: /etc/network/interfaces

```
allow-hotplug eth0
iface eth0 inet dhcp
auto eth1
iface eth1 inet static
address 172.16.2.28
netmask 255.255.255.0
```

Cuadro (126) - Configuración de red para nbds10

Path: /etc/network/interfaces

```
allow-hotplug eth0
iface eth0 inet dhcp
auto eth1
iface eth1 inet static
address 172.16.2.29
netmask 255.255.255.0
```

Cuadro (127) - Configuración de red para nbds11

Path: /etc/network/interfaces

```
allow-hotplug eth0
iface eth0 inet dhcp
auto eth1
iface eth1 inet static
address 172.16.2.30
netmask 255.255.255.0
```

Cuadro (128) - Configuración de red para nbds12

Path: /etc/network/interfaces

```
allow-hotplug eth0
iface eth0 inet dhcp
auto eth1
iface eth1 inet static
address 172.16.2.31
netmask 255.255.255.0
```

APÉNDICE G. Configuraciones DRBD

Iniciemos viendo todas las opciones para “global_common.conf:

Cuadro (129) - Opciones para global_common.conf

PATH: /etc/drbd.d/global_common.conf

```
# DRBD is the result of over a decade of development by LINBIT.
# In case you need professional services for DRBD or have
# feature requests visit http://www.linbit.com

global {
            #minor-count count
            #  may be a number from 1 to 1048575.
            #
            #  Minor-count is a sizing hint for DRBD. It helps to right-
size various memory pools.
            #  It should be set in the in the same order of magnitude
than the actual number of
            #  minors you use. Per default the module loads with 11 more
resources than you have
            #  currently in your config but at least 32.

            #dialog-refresh time
            #  may be 0 or a positive number.
            #
            #   The user dialog redraws the second count every time
seconds (or does no redraws if
            #  time is 0). The default value is 1.

            #disable-ip-verification
            #  Use disable-ip-verification if, for some obscure reasons,
drbdadm can/might not use
            #  ip or ifconfig to do a sanity check for the IP address.
You can disable the IP
            #  verification with this option.

            #usage-count val
            #   Please participate in DRBD's online usage counter[2].
The most convenient way to do
            #  so is to set this option to yes. Valid options are: yes,
no and ask.
}

common {
      handlers {
            # These are EXAMPLE handlers only.
            # They may have severe implications,
            # like hard resetting the node under certain circumstances.
            # Be careful when chosing your poison.
```

```
            #   pri-on-incon-degr   "/usr/lib/drbd/notify-pri-on-incon-
degr.sh;    /usr/lib/drbd/notify-emergency-reboot.sh;    echo    b    >
/proc/sysrq-trigger ; reboot -f";
            #  pri-lost-after-sb  "/usr/lib/drbd/notify-pri-lost-after-
sb.sh; /usr/lib/drbd/notify-emergency-reboot.sh; echo b > /proc/sysrq-
trigger ; reboot -f";
            #     local-io-error     "/usr/lib/drbd/notify-io-error.sh;
/usr/lib/drbd/notify-emergency-shutdown.sh;  echo  o  >  /proc/sysrq-
trigger ; halt -f";
            # fence-peer "/usr/lib/drbd/crm-fence-peer.sh";
            # split-brain "/usr/lib/drbd/notify-split-brain.sh root";
            # out-of-sync "/usr/lib/drbd/notify-out-of-sync.sh root";
            #   before-resync-target   "/usr/lib/drbd/snapshot-resync-
target-lvm.sh -p 15 -- -c 16k";
            #   after-resync-target   /usr/lib/drbd/unsnapshot-resync-
target-lvm.sh;
      }

      startup {
          #wfc-timeout time
        #  Wait for connection timeout.  The init script drbd(8) blocks
the
        #  boot process until the DRBD resources are connected. When
the
        #  cluster manager starts later, it does not see a resource
with
        #  internal split-brain. In case you want to limit the wait
time, do
        #  it here. Default is 0, which means unlimited. The unit is
seconds.

          #degr-wfc-timeout time
          #  Wait for connection timeout, if this node was a degraded
cluster.
          #  In case a degraded cluster (= cluster with only one node
left) is
          #   rebooted, this timeout value is used instead of wfc-
timeout,
          #  because the peer is less likely to show up in time, if it
had been
          #  dead before. Value 0 means unlimited.

          #outdated-wfc-timeout time
          #  Wait for connection timeout, if the peer was outdated. In
case a
          #  degraded cluster (= cluster with only one node left) with
an
          #  outdated peer disk is rebooted, this timeout value is used
instead
          #  of wfc-timeout, because the peer is not allowed to become
primary
          #  in the meantime. Value 0 means unlimited.

          #wait-after-sb
```

```
          #  By setting this option you can make the init script to
continue to
          #  wait even if the device pair had a split brain situation
and
          #  therefore refuses to connect.

      }

      options {
            #cpu-mask cpu-mask
            #  Sets the cpu-affinity-mask for DRBD's kernel threads of
this
            #  device. The default value of cpu-mask is 0, which means
that DRBD's
            #  kernel threads should be spread over all CPUs of the
machine. This
            #  value must be given in hexadecimal notation. If it is
too big it
            #  will be truncated.

            #on-no-data-accessible ond-policy
            #  This setting controls what happens to IO requests on a
degraded,
            #  disk less node (I.e. no data store is reachable). The
available
            #  policies are io-error and suspend-io.
            #
            #  If ond-policy is set to suspend-io you can either resume
IO by
            #  attaching/connecting the last lost data storage, or by
the drbdadm
            #  resume-io res command. The latter will result in IO
errors of
            #  course.
            #
            #  The default is io-error. This setting is available since
DRBD
            #  8.3.9.

      }

      disk {

            #size size
            #  You can override DRBD's size determination method with
this option.
            #  If you need to use the device before it was ever connected
to its
            #  peer, use this option to pass the size of the DRBD device
to the
            #  driver. Default unit is sectors (1s = 512 bytes).
            #
            #  If you use the size parameter in drbd.conf, we strongly
recommend
```

```
            #  to add an explicit unit postfix. drbdadm and drbdsetup
used to have
            #  mismatching default units.

            #on-io-error handler
            #  is taken, if the lower level device reports io-errors to
the upper
            #  layers.
            #
            #  handler may be pass_on, call-local-io-error or detach.
            #
            #    pass_on:  The  node  downgrades  the  disk  status  to
inconsistent, marks
            #  the erroneous block as inconsistent in the bitmap and
retries the
            #  IO on the remote node.
            #
            #  call-local-io-error: Call the handler script local-io-
error.
            #
            # detach: The node drops its low level device, and continues
in
            #  diskless mode.

            #fencing fencing_policy
            #  By fencing we understand preventive measures to avoid
situations
            #  where both nodes are primary and disconnected (AKA split
brain).
            #
            #  Valid fencing policies are:
            #
            #  dont-care
            #      This is the default policy. No fencing actions are
taken.
            #
            #  resource-only
            #      If a node becomes a disconnected primary, it tries
to fence the
            #      peer's disk. This is done by calling the fence-peer
handler.
            #      The handler is supposed to reach the other node over
            #       alternative communication paths and call 'drbdadm
outdate res'
            #      there.
            #
            #  resource-and-stonith
            #      If a node becomes a disconnected primary, it freezes
all its IO
            #       operations and calls its fence-peer handler. The
fence-peer
            #    handler is supposed to reach the peer over alternative
            #       communication paths and call 'drbdadm outdate res'
there. In
```

```
            #      case it cannot reach the peer it should stonith the
peer. IO is
            #     resumed as soon as the situation is resolved. In case
your
            #      handler fails, you can resume IO with the resume-io
command.

            #disk-barrier,
            #disk-flushes,
            #disk-drain
            #   DRBD has four implementations to express write-after-
write
            #   dependencies to its backing storage device. DRBD will
use the first
            #   method that is supported by the backing storage device
and that is
            #   not disabled. By default the flush method is used.
            #
            #    Since drbd-8.4.2 disk-barrier is disabled by default
because since
            #   linux-2.6.36 (or 2.6.32 RHEL6) there is no reliable way
to
            #    determine if queuing of IO-barriers works.  Dangerous
only enable
            #   if you are told so by one that knows for sure.
            #
            #   When selecting the method you should not only base your
decision on
            #   the measurable performance. In case your backing storage
device has
            #   a volatile write cache (plain disks, RAID of plain disks)
you
            #    should use one of the first two. In case your backing
storage
            #    device has battery-backed write cache you may go with
option 3.
            #   Option 4 (disable everything, use "none") is dangerous
on most IO
            #   stacks, may result in write-reordering, and if so, can
            #    theoretically be the reason for data corruption, or
disturb the
            #    DRBD protocol, causing spurious disconnect/reconnect
cycles.  Do
            #   not useno-disk-drain.
            #
            #    Unfortunately device mapper (LVM) might not support
barriers.
            #
            #    The letter after "wo:" in /proc/drbd indicates with
method is
            #     currently in use for a device: b, f, d, n. The
implementations are:
            #
            #   barrier
```

```
            #       The first requires that the driver of the backing
storage
            #        device support barriers (called 'tagged command
queuing' in
            #       SCSI and 'native command queuing' in SATA speak).
The use of
            #        this method can be enabled by setting the disk-
barrier options
            #      to yes.
            #
            #  flush
            #      The second requires that the backing device support
disk
            #       flushes (called 'force unit access' in the drive
vendors
            #     speak). The use of this method can be disabled setting
            #      disk-flushes to no.
            #
            #  drain
            #       The third method is simply to let write requests
drain before
            #     write requests of a new reordering domain are issued.
This was
            #      the only implementation before 8.0.9.
            #
            #  none
            #       The fourth method is to not express write-after-
write
            #       dependencies to the backing store at all, by also
specifying
            #      no-disk-drain. This is dangerous on most IO stacks,
may result
            #      in write-reordering, and if so, can theoretically be
the reason
            #       for data corruption, or disturb the DRBD protocol,
causing
            #      spurious disconnect/reconnect cycles.  Do not useno-
disk-drain.

            #md-flushes
            #  Disables the use of disk flushes and barrier BIOs when
accessing
            #  the meta data device. See the notes on disk-flushes.

            #resync-rate rate
            #  To ensure a smooth operation of the application on top
of DRBD, it
            #  is possible to limit the bandwidth which may be used by
background
            #  synchronizations. The default is 250 KB/sec, the default
unit is
            #  KB/sec. Optional suffixes K, M, G are allowed.

            #resync-after res-name
```

```
            #  By default, resynchronization of all devices would run
in parallel.
            #     By   defining   a   resync-after   dependency,   the
resynchronization of
            #  this resource will start only if the resource res-name
is already
            #     in   connected   state   (i.e.,   has   finished   its
resynchronization).

            #al-extents extents
            #  DRBD automatically performs hot area detection. With this
parameter
            #  you control how big the hot area (= active set) can get.
Each
            #   extent marks 4M of the backing storage (= low-level
device). In
            #  case a primary node leaves the cluster unexpectedly, the
areas
            #  covered by the active set must be resynced upon rejoining
of the
            #   failed node. The data structure is stored in the meta-
data area,
            #   therefore each change of the active set is a write
operation to the
            #  meta-data device. A higher number of extents gives longer
resync
            #   times but less updates to the meta-data. The default
number of
            #  extents is 1237. (Minimum: 7, Maximum: 65534)
            #
            #  Note that the effective maximum may be smaller, depending
on how
            #  you created the device meta data, see also drbdmeta(8).
The
            #  effective maximum is 919 * (available on-disk activity-
log
            #   ring-buffer area/4kB -1), the default 32kB ring-buffer
effects a
            #   maximum of 6433 (covers more than 25 GiB of data). We
recommend to
            #   keep this well within the amount your backend storage
and
            #   replication link are able to resync inside of about 5
minutes.

            #c-plan-ahead plan_time,
            #c-fill-target fill_target,
            #c-delay-target delay_target,
            #c-max-rate max_rate
            #   The dynamic resync speed controller gets enabled with
setting
            #  plan_time to a positive value. It aims to fill the buffers
along
```

```
            #   the data path with either a constant amount of data
fill_target, or
            #  aims to have a constant delay time of delay_target along
the path.
            #  The controller has an upper bound of max_rate.
            #
            #  By plan_time the agility of the controller is configured.
Higher
            #  values yield for slower/lower responses of the controller
to
            #   deviation from the target value. It should be at least
5 times RTT.
            #   For regular data paths a fill_target in the area of 4k
to 100k is
            #  appropriate. For a setup that contains drbd-proxy it is
advisable
            #   to use delay_target instead. Only when fill_target is
set to 0 the
            #    controller will use delay_target. 5 times RTT is a
reasonable
            #  starting value.  Max_rate should be set to the bandwidth
available
            #   between the DRBD-hosts and the machines hosting DRBD-
proxy, or to
            #  the available disk-bandwidth.
            #
            #  The default value of plan_time is 0, the default unit is
0.1
            #  seconds.  Fill_target has 0 and sectors as default unit.
            #   Delay_target has 1 (100ms) and 0.1 as default unit.
Max_rate has
            #  10240 (100MiB/s) and KiB/s as default unit.
            #
            #  The dynamic resync speed controller and its settings are
available
            #  since DRBD 8.3.9.

            #c-min-rate min_rate
            #  A node that is primary and sync-source has to schedule
application
            #  IO requests and resync IO requests. The min_rate tells
DRBD use
            #   only up to min_rate for resync IO and to dedicate all
other
            #  available IO bandwidth to application requests.
            #
            #  Note: The value 0 has a special meaning. It disables the
limitation
            #    of  resync  IO  completely,  which  might  slow  down
application IO
            #  considerably. Set it to a value of 1, if you prefer that
resync IO
            #  never slows down application IO.
            #
```

```
            #  Note: Although the name might suggest that it is a lower
bound for
            #  the dynamic resync speed controller, it is not. If the
DRBD-proxy
            #  buffer is full, the dynamic resync speed controller is
free to
            #  lower the resync speed down to 0, completely independent
of the
            #  c-min-rate setting.
            #
            #  Min_rate has 4096 (4MiB/s) and KiB/s as default unit.

            #disk-timeout
            #  If the lower-level device on which a DRBD device stores
its data
            #  does not finish an I/O request within the defined disk-
timeout,
            #  DRBD treats this as a failure. The lower-level device is
detached,
            #  and the device's disk state advances to Diskless. If DRBD
is
            #   connected to one or more peers, the failed request is
passed on to
            #  one of them.
            #
            #  This option is dangerous and may lead to kernel panic!
            #
            #   "Aborting" requests, or force-detaching the disk, is
intended for
            #  completely blocked/hung local backing devices which do
no longer
            #  complete requests at all, not even do error completions.
In this
            #  situation, usually a hard-reset and failover is the only
way out.
            #
            #  By "aborting", basically faking a local error-completion,
we allow
            #   for a more graceful swichover by cleanly migrating
services. Still
            #  the affected node has to be rebooted "soon".
            #
            #  By completing these requests, we allow the upper layers
to re-use
            #  the associated data pages.
            #
            #   If later the local backing device "recovers", and now
DMAs some
            #  data from disk into the original request pages, in the
best case it
            #  will just put random data into unused pages; but typically
it will
            #  corrupt meanwhile completely unrelated data, causing all
sorts of
```

```
            #  damage.
            #
            #   Which means delayed successful completion, especially
for READ
            #  requests, is a reason to panic(). We assume that a delayed
*error*
            #  completion is OK, though we still will complain noisily
about it.
            #
            #  The default value of disk-timeout is 0, which stands for
an
            #  infinite timeout. Timeouts are specified in units of 0.1
seconds.
            #  This option is available since DRBD 8.3.12.

      }

      net {

            #timeout time
            #  If the partner node fails to send an expected response
packet within time tenths of
            #   a second, the partner node is considered dead and
therefore the TCP/IP connection
            #  is abandoned. This must be lower than connect-int and
ping-int. The default value
            #  is 60 = 6 seconds, the unit 0.1 seconds.

            #max-epoch-size number
            #   The highest number of data blocks between two write
barriers. If you set this
            #  smaller than 10, you might decrease your performance.

            #max-buffers number
            #  Limits the memory usage per DRBD minor device on the
receiving side, or for
            #  internal buffers during resync or online-verify. Unit is
PAGE_SIZE, which is 4 KiB
            #  on most systems. The minimum possible setting is hard
coded to 32 (=128 KiB). These
            #   buffers are used to hold data blocks while they are
written to/read from disk. To
            #  avoid possible distributed deadlocks on congestion, this
setting is used as a
            #  throttle threshold rather than a hard limit. Once more
than max-buffers pages are
            #  in use, further allocation from this pool is throttled.
You want to increase
            #  max-buffers if you cannot saturate the IO backend on the
receiving side.

            #unplug-watermark number
            #  This setting has no effect with recent kernels that use
explicit on-stack plugging
```

```
            #   (upstream Linux kernel 2.6.39, distributions may have
backported).

            #  When the number of pending write requests on the standby
(secondary) node exceeds
            #  the unplug-watermark, we trigger the request processing
of our backing storage
            #    device. Some storage controllers deliver better
performance with small values,
            #  others deliver best performance when the value is set to
the same value as
            #  max-buffers, yet others don't feel much effect at all.
Minimum 16, default 128,
            #  maximum 131072.

            #connect-int time
            #  In case it is not possible to connect to the remote DRBD
device immediately, DRBD
            #  keeps on trying to connect. With this option you can set
the time between two
            #  retries. The default value is 10 seconds, the unit is 1
second.

            #ping-int time
            #  If the TCP/IP connection linking a DRBD device pair is
idle for more than time
            #  seconds, DRBD will generate a keep-alive packet to check
if its partner is still
            #  alive. The default is 10 seconds, the unit is 1 second.

            #sndbuf-size size
            #  is the size of the TCP socket send buffer. The default
value is 0, i.e. autotune.
            #  You can specify smaller or larger values. Larger values
are appropriate for
            #   reasonable write throughput with protocol A over high
latency networks. Values
            #  below 32K do not make sense. Since 8.0.13 resp. 8.2.7,
setting the size value to 0
            #  means that the kernel should autotune this.

            #rcvbuf-size size
            #  is the size of the TCP socket receive buffer. The default
value is 0, i.e.
            #   autotune. You can specify smaller or larger values.
Usually this should be left at
            #  its default. Setting the size value to 0 means that the
kernel should autotune
            #  this.

            #ko-count number
            #  In case the secondary node fails to complete a single
write request for count times
```

```
            #  the timeout, it is expelled from the cluster. (I.e. the
primary node goes into
            #  StandAlone mode.) The default value is 0, which disables
this feature.

            #allow-two-primaries
            #  With this option set you may assign the primary role to
both nodes. You only should
            #  use this option if you use a shared storage file system
on top of DRBD. At the time
            #  of writing the only ones are: OCFS2 and GFS. If you use
this option with any other
            #  file system, you are going to crash your nodes and to
corrupt your data!

            #cram-hmac-alg
            #  You need to specify the HMAC algorithm to enable peer
authentication at all. You
            #  are strongly encouraged to use peer authentication. The
HMAC algorithm will be used
            #  for the challenge response authentication of the peer.
You may specify any digest
            #  algorithm that is named in /proc/crypto.

            #shared-secret
            #  The shared secret used in peer authentication. May be up
to 64 characters. Note
            #  that peer authentication is disabled as long as no cram-
hmac-alg (see above) is
            #  specified.

            #after-sb-0pri  policy
            #  possible policies are:
            #
            #  disconnect
            #      No automatic resynchronization, simply disconnect.
            #
            #  discard-younger-primary
            #      Auto sync from the node that was primary before the
split-brain situation
            #      happened.
            #
            #  discard-older-primary
            #     Auto sync from the node that became primary as second
during the split-brain
            #      situation.
            #
            #  discard-zero-changes
            #       In case one node did not write anything since the
split brain became evident,
            #      sync from the node that wrote something to the node
that did not write
            #       anything. In case none wrote anything this policy
uses a random decision to
```

```
            #       perform a "resync" of 0 blocks. In case both have
written something this policy
            #       disconnects the nodes.
            #
            #  discard-least-changes
            #       Auto sync from the node that touched more blocks
during the split brain
            #       situation.
            #
            #  discard-node-NODENAME
            #       Auto sync to the named node.

            #after-sb-1pri  policy
            #  possible policies are:
            #
            #  disconnect
            #       No automatic resynchronization, simply disconnect.
            #
            #  consensus
            #       Discard the version of the secondary if the outcome
of the after-sb-0pri
            #      algorithm would also destroy the current secondary's
data. Otherwise
            #       disconnect.
            #
            #  violently-as0p
            #         Always take the decision of the after-sb-0pri
algorithm, even if that causes an
            #       erratic change of the primary's view of the data.
This is only useful if you
            #       use a one-node FS (i.e. not OCFS2 or GFS) with the
allow-two-primaries flag,
            #       AND if you really know what you are doing. This is
DANGEROUS and MAY CRASH YOUR
            #        MACHINE if you have an FS mounted on the primary
node.
            #
            #  discard-secondary
            #       Discard the secondary's version.
            #
            #  call-pri-lost-after-sb
            #         Always honor the outcome of the after-sb-0pri
algorithm. In case it decides the
            #       current secondary has the right data, it calls the
"pri-lost-after-sb" handler
            #       on the current primary.

            #after-sb-2pri  policy
            #  possible policies are:
            #
            #  disconnect
            #       No automatic resynchronization, simply disconnect.
            #
            #  violently-as0p
```

```
            #         Always take the decision of the after-sb-0pri
algorithm, even if that causes an
            #       erratic change of the primary's view of the data.
This is only useful if you
            #      use a one-node FS (i.e. not OCFS2 or GFS) with the
allow-two-primaries flag,
            #      AND if you really know what you are doing. This is
DANGEROUS and MAY CRASH YOUR
            #       MACHINE if you have an FS mounted on the primary
node.
            #
            #  call-pri-lost-after-sb
            #      Call the "pri-lost-after-sb" helper program on one
of the machines. This
            #     program is expected to reboot the machine, i.e. make
it secondary.

            #always-asbp
            #   Normally the automatic after-split-brain policies are
only used if current states
            #   of the UUIDs do not indicate the presence of a third
node.
            #
            #  With this option you request that the automatic after-
split-brain policies are used
            #  as long as the data sets of the nodes are somehow related.
This might cause a full
            #  sync, if the UUIDs indicate the presence of a third node.
(Or double faults led to
            #  strange UUID sets.)

            #rr-conflict  policy
            #  This option helps to solve the cases when the outcome of
the resync decision is
            #   incompatible with the current role assignment in the
cluster.
            #
            #  disconnect
            #      No automatic resynchronization, simply disconnect.
            #
            #  violently
            #      Sync to the primary node is allowed, violating the
assumption that data on a
            #        block device are stable for one of the nodes.
Dangerous, do not use.
            #
            #  call-pri-lost
            #       Call the "pri-lost" helper program on one of the
machines. This program is
            #         expected to reboot the machine, i.e. make it
secondary.

            #ping-timeout time
```

```
            #  The time the peer has time to answer to a keep-alive
packet. In case the peer's
            #  reply is not received within this time period, it is
considered as dead. The
            #  default value is 500ms, the default unit are tenths of
a second.

            #data-integrity-alg  alg
            #  DRBD can ensure the data integrity of the user's data on
the network by comparing
            #   hash values. Normally this is ensured by the 16 bit
checksums in the headers of
            #  TCP/IP packets.
            #
            #  This option can be set to any of the kernel's data digest
algorithms. In a typical
            #  kernel configuration you should have at least one of md5,
sha1, and crc32c
            #  available. By default this is not enabled.
            #
            #  See also the notes on data integrity.

            #tcp-cork
            #  DRBD usually uses the TCP socket option TCP_CORK to hint
to the network stack when
            #  it can expect more data, and when it should flush out
what it has in its send
            #  queue. It turned out that there is at least one network
stack that performs worse
            #   when one uses this hinting method. Therefore we
introducted this option. By setting
            #  tcp-cork to no you can disable the setting and clearing
of the TCP_CORK socket
            #  option by DRBD.

            #on-congestion congestion_policy,
            #congestion-fill fill_threshold,
            #congestion-extents active_extents_threshold
            #  By default DRBD blocks when the available TCP send queue
becomes full. That means
            #  it will slow down the application that generates the
write requests that cause DRBD
            #  to send more data down that TCP connection.
            #
            #  When DRBD is deployed with DRBD-proxy it might be more
desirable that DRBD goes
            #  into AHEAD/BEHIND mode shortly before the send queue
becomes full. In AHEAD/BEHIND
            #  mode DRBD does no longer replicate data, but still keeps
the connection open.
            #
            #  The advantage of the AHEAD/BEHIND mode is that the
application is not slowed down,
```

```
            #  even if DRBD-proxy's buffer is not sufficient to buffer
all write requests. The
            #  downside is that the peer node falls behind, and that a
resync will be necessary to
            #  bring it back into sync. During that resync the peer node
will have an inconsistent
            #  disk.
            #
            #   Available congestion_policys are block and pull-ahead.
The default is block.
            #  Fill_threshold might be in the range of 0 to 10GiBytes.
The default is 0 which
            #   disables the check.  Active_extents_threshold has the
same limits as al-extents.
            #
            #   The AHEAD/BEHIND mode and its settings are available
since DRBD 8.3.10.

            #verify-alg hash-alg
            #  During online verification (as initiated by the verify
sub-command), rather than
            #  doing a bit-wise comparison, DRBD applies a hash function
to the contents of every
            #  block being verified, and compares that hash with the
peer. This option defines the
            #  hash algorithm being used for that purpose. It can be
set to any of the kernel's
            #  data digest algorithms. In a typical kernel configuration
you should have at least
            #  one of md5, sha1, and crc32c available. By default this
is not enabled; you must
            #  set this option explicitly in order to be able to use
on-line device verification.
            #
            #  See also the notes on data integrity.

            #csums-alg hash-alg
            #  A resync process sends all marked data blocks from the
source to the destination
            #   node, as long as no csums-alg is given. When one is
specified the resync process
            #  exchanges hash values of all marked blocks first, and
sends only those data blocks
            #  that have different hash values.
            #
            #  This setting is useful for DRBD setups with low bandwidth
links. During the restart
            #   of a crashed primary node, all blocks covered by the
activity log are marked for
            #  resync. But a large part of those will actually be still
in sync, therefore using
            #  csums-alg will lower the required bandwidth in exchange
for CPU cycles.
```

```
            #use-rle
            #  During resync-handshake, the dirty-bitmaps of the nodes
are exchanged and merged
            #   (using bit-or), so the nodes will have the same
understanding of which blocks are
            #  dirty. On large devices, the fine grained dirty-bitmap
can become large as well,
            #  and the bitmap exchange can take quite some time on low-
bandwidth links.
            #
            #  Because the bitmap typically contains compact areas where
all bits are unset
            #   (clean) or set (dirty), a simple run-length encoding
scheme can considerably reduce
            #  the network traffic necessary for the bitmap exchange.
            #
            #  For backward compatibilty reasons, and because on fast
links this possibly does not
            #   improve transfer time but consumes cpu cycles, this
defaults to off.

      }
}
```

Cuadro (130) - Archivo de configuración global_common.conf en nbds3

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}

common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
      }

      disk {
…
      }

      net {
        protocol C;
      }
```

```
}
```

Cuadro (131) - Archivo de configuración r0.res en nbds3

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds3 {
    address   172.16.2.22:7789;
  }
  on nbds4 {
    address   172.16.2.23:7789;
  }
}
```

Cuadro (132) - Archivo de configuración global_common.conf en nbds4

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}

common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
      }

      disk {
…
      }

      net {
        protocol C;
      }
}
```

Cuadro (133) - Archivo de configuración r0.res en nbds4

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds3 {
    address   172.16.2.22:7789;
  }
  on nbds4 {
    address   172.16.2.23:7789;
  }
}
```

Cuadro (134) - Archivo de configuración global_common.conf en nbds5

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}

common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
      }

      disk {
…
      }

      net {
        protocol C;
      }
}
```

Cuadro (135) - Archivo de configuración r0.res en nbds5

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds5 {
    address   172.16.2.24:7789;
  }
  on nbds6 {
    address   172.16.2.25:7789;
  }
}
```

Cuadro (136) - Archivo de configuración global_common.conf en nbds6

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}

common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
      }

      disk {
…
      }

      net {
        protocol C;
      }
}
```

Cuadro (137) - Archivo de configuración r0.res en nbds6

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds5 {
    address   172.16.2.24:7789;
  }
  on nbds6 {
    address   172.16.2.25:7789;
  }
}
```

Cuadro (138) - Archivo de configuración global_common.conf en nbds7

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}

common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
      }

      disk {
…
      }

      net {
        protocol C;
      }
}
```

Cuadro (139) - Archivo de configuración r0.res en nbds7

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds7 {
    address   172.16.2.26:7789;
  }
  on nbds8 {
    address   172.16.2.27:7789;
  }
}
```

Cuadro (140) - Archivo de configuración global_common.conf en nbds8

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}

common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
      }

      disk {
…
      }

      net {
        protocol C;
      }
}
```

Cuadro (141) - Archivo de configuración r0.res en nbds8

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds7 {
    address   172.16.2.26:7789;
  }
  on nbds8 {
    address   172.16.2.27:7789;
  }
}
```

Cuadro (142) - Archivo de configuración global_common.conf en nbds9

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}

common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
      }

      disk {
…
      }

      net {
        protocol C;
      }
}
```

Cuadro (143) - Archivo de configuración r0.res en nbds9

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds7 {
    address   172.16.2.28:7789;
  }
  on nbds8 {
    address   172.16.2.29:7789;
  }
}
```

Cuadro (144) - Archivo de configuración global_common.conf en nbds10

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}

common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
      }

      disk {
…
      }

      net {
        protocol C;
      }
}
```

Cuadro (145) - Archivo de configuración r0.res en nbds10

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds7 {
    address   172.16.2.28:7789;
  }
  on nbds8 {
    address   172.16.2.29:7789;
  }
}
```

Cuadro (146) - Archivo de configuración global_common.conf en nbds11

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}

common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
      }

      disk {
…
      }

      net {
        protocol C;
      }
}
```

Cuadro (147) - Archivo de configuración r0.res en nbds11

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds7 {
    address   172.16.2.30:7789;
  }
  on nbds8 {
    address   172.16.2.31:7789;
  }
}
```

Cuadro (148) - Archivo de configuración global_common.conf en nbds12

Path: /etc/drbd.d/global_common.conf

```
…

global {
      usage-count no;
…
}

common {
      handlers {
…
      }

      startup {
…
      }

      options {
…
      }

      disk {
…
      }

      net {
        protocol C;
      }
}
```

Cuadro (149) - Archivo de configuración r0.res en nbds12

Path: /etc/drbd.d/r0.res

```
resource r0 {
  device    /dev/drbd1;
  disk      /dev/sdb1;
  meta-disk internal;
  on nbds7 {
    address   172.16.2.30:7789;
  }
  on nbds8 {
    address   172.16.2.31:7789;
  }
}
```

APÉNDICE H. Configurando el servidor NBD

Cuadro (150) - Configuración del archivo config en nbds5

Path: /etc/nbd-server/config

```
[generic]
# If you want to run everything as root rather than the nbd user, you
# may either say "root" in the two following lines, or remove them
# altogether. Do not remove the [generic] section, however.
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true

# What follows are export definitions. You may create as much of them
as
# you want, but the section header has to be unique.
      [disk3]
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Cuadro (151) - Configuración del archivo allow en nbds5

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

Configurando el servidor nbds6:

Cuadro (152) - Configuración del archivo config en nbds6

Path: /etc/nbd-server/config

```
[generic]
# If you want to run everything as root rather than the nbd user, you
# may either say "root" in the two following lines, or remove them
# altogether. Do not remove the [generic] section, however.
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true

# What follows are export definitions. You may create as much of them
as
# you want, but the section header has to be unique.
      [disk3]
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Cuadro (153) - Configuración del archivo allow en nbds6

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

Configurando el servidor nbds7:

Cuadro (154) - Configuración del archivo config.en nbds7

Path: /etc/nbd-server/config

```
[generic]
# If you want to run eveurything as root rather than the nbd user, you
# may either say "root" in the two following lines, or remove them
# altogether. Do not remove the [generic] section, however.
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true

# What follows are export definitions. You may create as much of them
as
# you want, but the section header has to be unique.
      [disk4]
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Cuadro (155) - Configuración del archivo allow en nbds7

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

Configurando el servidor nbds8:

Cuadro (156) - Configuración del archivo config.en nbds8

Path: /etc/nbd-server/config

```
[generic]
# If you want to run everything as root rather than the nbd user, you
# may either say "root" in the two following lines, or remove them
# altogether. Do not remove the [generic] section, however.
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true

# What follows are export definitions. You may create as much of them
as
# you want, but the section header has to be unique.
      [disk4]
```

```
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Cuadro (157) - Configuración del archivo allow en nbds8

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

Configurando el servidor nbds9:

Cuadro (158) - Configuración del archivo config.en nbds9

Path: /etc/nbd-server/config

```
[generic]
# If you want to run everything as root rather than the nbd user, you
# may either say "root" in the two following lines, or remove them
# altogether. Do not remove the [generic] section, however.
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true

# What follows are export definitions. You may create as much of them
as
# you want, but the section header has to be unique.
      [disk5]
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Cuadro (159) - Configuración del archivo allow en nbds9

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

Configurando el servidor nbds10:

Cuadro (160) - Configuración del archivo config.en nbds10

Path: /etc/nbd-server/config

```
[generic]
# If you want to run everything as root rather than the nbd user, you
# may either say "root" in the two following lines, or remove them
# altogether. Do not remove the [generic] section, however.
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true
```

```
# What follows are export definitions. You may create as much of them
as
# you want, but the section header has to be unique.
      [disk5]
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Cuadro (161) - Configuración del archivo allow en nbds10

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

Configurando el servidor nbds11:

Cuadro (162) - Configuración del archivo config.en nbds11

Path: /etc/nbd-server/config

```
[generic]
# If you want to run everything as root rather than the nbd user, you
# may either say "root" in the two following lines, or remove them
# altogether. Do not remove the [generic] section, however.
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true

# What follows are export definitions. You may create as much of them
as
# you want, but the section header has to be unique.
      [disk6]
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Cuadro (163) - Configuración del archivo allow en nbds11

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

Configurando el servidor nbds12:

Cuadro (164) - Configuración del archivo config.en nbds12

Path: /etc/nbd-server/config

```
[generic]
# If you want to run everything as root rather than the nbd user, you
# may either say "root" in the two following lines, or remove them
```

```
# altogether. Do not remove the [generic] section, however.
      user = nbd
      group = nbd
      includedir = /etc/nbd-server/conf.d
      allowlist = true

# What follows are export definitions. You may create as much of them
as
# you want, but the section header has to be unique.
      [disk6]
      authfile=/etc/nbd-server/allow
      exportname=/dev/drbd1
```

Cuadro (165) - Configuración del archivo allow en nbds12

Path: /etc/nbd-server/allow

```
172.16.2.10/32
172.16.2.11/32
172.16.2.15/32
```

APÉNDICE I. Configuraciones para HEARTBEAT

Cuadro (166) - Archivo de configuración authkeys en nbds3

Path: /etc/ha.d/haresources

```
auth 1
1 sha1 (stdin)= 919beeecd430dc1a74de557830325607
```

Cuadro (167) - Archivo de configuración ha.cf en nbds3

Path: /etc/ha.d/ha.cf

```
…
debugfile /var/log/ha.debug
…
logfile /var/log/ha.log
…
logfacility     local0
…
keepalive 1
…

deadtime 15
…
warntime 10
…
initdead 30
…
udpport 694
…
ucast eth0 172.16.2.23
…
auto_failback off
…
node nbds3
node nbds4
…
```

Cuadro (168) - Archivo de configuración haresources en nbds3

Path: /etc/ha.d/haresources

```
nbds3 drbddisk::r0 172.16.2.51 nbd-server
```

Cuadro (169) - Archivo de configuración authkeys en nbds4

Path: /etc/ha.df/authkeys

```
auth 1
1 sha1 (stdin)= 919beeecd430dc1a74de557830325607
```

Cuadro (170) - Archivo de configuración ha.cf en nbds4

Path: /etc/ha.d/ha.cf

```
debugfile /var/log/ha.debug
…
logfile /var/log/ha.log
…
logfacility     local0
…
keepalive 1
…
deadtime 15
…
warntime 10
…
initdead 30
…
udpport 694
…
ucast eth0 172.16.2.22
…
auto_failback off
…
node nbds3
node nbds4
```

Cuadro (171) - Archivo de configuración haresources en nbds4

Path: /etc/ha.d/haresources

```
nbds3 drbddisk::r0 172.16.2.51 nbd-server
```

Cuadro (172) - Archivo de configuración authkeys en nbds5

Path: /etc/ha.df/authkeys

```
auth 1
1 sha1 (stdin)= 8ac577df8fadcd0167614865a5dc1c8f
```

Cuadro (173) - Archivo de configuración ha.cf en nbds5

Path: /etc/ha.d/ha.cf

```
debugfile /var/log/ha.debug
…
logfile /var/log/ha.log
…
logfacility     local0
…
keepalive 1
…
deadtime 15
…
warntime 10
…
```

```
initdead 30
…
udpport 694
…
ucast eth0 172.16.2.25
…
auto_failback off
…
node nbds5
node nbds6
```

Cuadro (174) - Archivo de configuración haresources en nbds5

Path: /etc/ha.d/haresources

```
nbds5 drbddisk::r0 172.16.2.52 nbd-server
```

Cuadro (175) - Archivo de configuración authkeys en nbds6

Path: /etc/ha.d/authkeys

```
auth 1
1 sha1 (stdin)= 8ac577df8fadcd0167614865a5dc1c8f
```

Cuadro (176) - Archivo de configuración ha.cf en nbds6

Path: /etc/ha.d/ha.cf

```
debugfile /var/log/ha.debug
…
logfile /var/log/ha.log
…
logfacility     local0
…
keepalive 1
…
deadtime 15
…
warntime 10
…
initdead 30
…
udpport 694
…
ucast eth0 172.16.2.24
…
auto_failback off
…
node nbds5
node nbds6
```

Cuadro (177) - Archivo de configuración haresources en nbds6

Path: /etc/ha.d/haresources

```
nbds5 drbddisk::r0 172.16.2.52 nbd-server
```

Cuadro (178) - Archivo de configuración authkeys en nbds7

Path: /etc/ha.d/authkeys

```
auth 1
1 sha1 (stdin)= 8ac577df8fadcd0167614865a5dc1c8f
```

Cuadro (179) - Archivo de configuración ha.cf en nbds7

Path: /etc/ha.d/ha.cf

```
debugfile /var/log/ha.debug
…
logfile /var/log/ha.log
…
logfacility     local0
…
keepalive 1
…
deadtime 15
…
warntime 10
…
initdead 30
…
udpport 694
…
ucast eth0 172.16.2.27
…
auto_failback off
…
node nbds7
node nbds8
```

Cuadro (180) - Archivo de configuración haresources en nbds7

Path: /etc/ha.d/haresources

```
nbds7 drbddisk::r0 172.16.2.53 nbd-server
```

Cuadro (181) - Archivo de configuración authkeys en nbds8

Path: /etc/ha.d/authkeys

```
auth 1
1 sha1 (stdin)= 8ac577df8fadcd0167614865a5dc1c8f
```

Cuadro (182) - Archivo de configuración ha.cf en nbds8

Path: /etc/ha.d/ha.cf

```
debugfile /var/log/ha.debug
…
logfile /var/log/ha.log
…
logfacility     local0
…
keepalive 1
…
deadtime 15
…
warntime 10
…
initdead 30
…
udpport 694
…
ucast eth0 172.16.2.26
…
auto_failback off
…
node nbds7
node nbds8
```

Cuadro (183) - Archivo de configuración haresources en nbds8

Path: /etc/ha.d/haresources

```
nbds7 drbddisk::r0 172.16.2.53 nbd-server
```

Cuadro (184) - Archivo de configuración authkeys en nbds9

Path: /etc/ha.d/authkeys

```
auth 1
1 sha1 (stdin)= 8ac577df8fadcd0167614865a5dc1c8f
```

Cuadro (185) - Archivo de configuración ha.cf en nbds9

Path: /etc/ha.d/ha.cf

```
debugfile /var/log/ha.debug
…
logfile /var/log/ha.log
…
logfacility     local0
…
keepalive 1
…
deadtime 15
…
warntime 10
…
```

```
initdead 30
…
udpport 694
…
ucast eth0 172.16.2.29
…
auto_failback off
…
node nbds9
node nbds10
```

Cuadro (186) - Archivo de configuración haresources en nbds9

Path: /etc/ha.d/haresources

```
nbds9 drbddisk::r0 172.16.2.54 nbd-server
```

Cuadro (187) - Archivo de configuración authkeys en nbds10

Path: /etc/ha.d/authkeys

```
auth 1
1 sha1 (stdin)= 8ac577df8fadcd0167614865a5dc1c8f
```

Cuadro (188) - Archivo de configuración ha.cf en nbds10

Path: /etc/ha.d/ha.cf

```
debugfile /var/log/ha.debug
…
logfile /var/log/ha.log
…
logfacility     local0
…
keepalive 1
…
deadtime 15
…
warntime 10
…
initdead 30
…
udpport 694
…
ucast eth0 172.16.2.28
…
auto_failback off
…
node nbds7
node nbds8
```

Cuadro (189) - Archivo de configuración haresources en nbds10

Path: /etc/ha.d/haresources

```
nbds9 drbddisk::r0 172.16.2.54 nbd-server
```

Cuadro (190) - Archivo de configuración authkeys en nbds11

Path: /etc/ha.d/authkeys

```
auth 1
1 sha1 (stdin)= 8ac577df8fadcd0167614865a5dc1c8f
```

Cuadro (191) - Archivo de configuración ha.cf en nbds11

Path: /etc/ha.d/ha.cf

```
debugfile /var/log/ha.debug
…
logfile /var/log/ha.log
…
logfacility     local0
…
keepalive 1
…
deadtime 15
…
warntime 10
…
initdead 30
…
udpport 694
…
ucast eth0 172.16.2.31
…
auto_failback off
…
node nbds7
node nbds8
```

Cuadro (192) - Archivo de configuración haresources en nbds11

Path: /etc/ha.d/haresources

```
nbds11 drbddisk::r0 172.16.2.55 nbd-server
```

Cuadro (193) - Archivo de configuración authkeys en nbds12

Path: /etc/ha.d/authkeys

```
auth 1
1 sha1 (stdin)= 8ac577df8fadcd0167614865a5dc1c8f
```

Cuadro (194) - Archivo de configuración ha.cf en nbds12

Path: /etc/ha.d/ha.cf

```
debugfile /var/log/ha.debug
…
logfile /var/log/ha.log
…
logfacility     local0
…
keepalive 1
…
deadtime 15
…
warntime 10
…
initdead 30
…
udpport 694
…
ucast eth0 172.16.2.30
…
auto_failback off
…
node nbds7
node nbds8
```

Cuadro (195) - Archivo de configuración haresources en nbds12

Path: /etc/ha.d/haresources

```
nbds11 drbddisk::r0 172.16.2.55 nbd-server
```

APÉNDICE J. SCRIPTS

Los Scripts detallados a continuación fueron desarrollados para la finalidad de encendido y apagado del servicio del arreglo. A cada uno de ellos se le diseño un helper que al ejecutarlo sin parámetros se muestra las opciones que tienen disponible.

Cuadro (196) - Script para arranque y detención del arreglo en servidores nbdc1 y nbdc2

Path: /etc/init.d/array_services

```
#!/bin/bash
### BEGIN INIT INFO
# Provides:          array_services
# Required-Start:    nbd-client
# Required-Stop:     nbd-client
# Should-Start:
# Should-Stop:
# X-Start-Before:
# X-Stop-After:
# Default-Start:     S
# Default-Stop:
### END INIT INFO


function start(){
  echo "Starting nbd"
  /etc/init.d/nbd-client start
  sleep 1
  #sleep 3
  echo "Starting mdadm"
  /etc/init.d/mdadm-raid start
  sleep 1
  echo "Assembling array"
  mdadm --assemble --scan
  sleep 1
  #sleep 3
  echo "Starting lvm"
  /root/bin/lvm start
  sleep 1
  mount /dev/VGA/vwh /srv/disk/
  #echo "Starting nfs"
  #/etc/init.d/nfs-kernel-server start
  #sleep 3
  sleep 1
  systemctl start vsftpd
}

function stop(){
```

```
  systemctl stop vsftpd
  sleep 1
  umount /srv/disk
  sleep 1
  #echo "Stopping nfs"
  #/etc/init.d/nfs-kernel-server stop
  echo "Stopping lvm"
  /root/bin/lvm stop
  sleep 1
  #sleep 3
  echo "Stopping mdadm"
  mdadm --stop /dev/md0
  sleep 1
  #sleep 3
  echo "Stopping nbd"
  /etc/init.d/nbd-client stop
}

case "$1" in
   start)
    start
    ;;
   stop)
    stop
    ;;
   restart)
    stop
    sleep 2
    start
    ;;
   *)
    echo "usage: $0 {start|stop|restart}"
    exit 1
    ;;
esac
```

Cuadro (197) - Scripts de arranque y detención de servicio lvm

Path: /root/bin/lvm

```
#!/bin/sh
### BEGIN INIT INFO
# Provides:          lvm
# Required-Start:    mountdevsubfs
# Required-Stop:
# Should-Start:      udev mdadm-raid cryptdisks-early multipath-tools-
boot
# Should-Stop:       umountroot mdadm-raid
# X-Start-Before:    checkfs mountall
# X-Stop-After:      umountfs
# Default-Start:     S
# Default-Stop:
### END INIT INFO

SCRIPTNAME=/etc/init.d/lvm
```

```
. /lib/lsb/init-functions

[ -x /sbin/vgchange ] || exit 0

start()
{
        echo "Setting up Logical Volume Management"
        /sbin/vgscan
        /sbin/vgchange -a y
        #log_action_end_msg "$?"
}

stop()
{
        echo "Stopping Logical Volume Management"
        /sbin/vgchange -a n
        log_action_end_msg "$?"
}

# See how we were called.


case "$1" in
   start)
    start
    ;;

   stop)
    stop
    ;;

   restart)
    stop
    sleep 2
    start
    ;;
   *)
    echo "usage: $0 {start|stop|restart}"
    exit 1
    ;;
esac
```

Cuadro (198) - Scripts de arranque y detención de servicio clúster

Path: /root/bin/stop_cluster

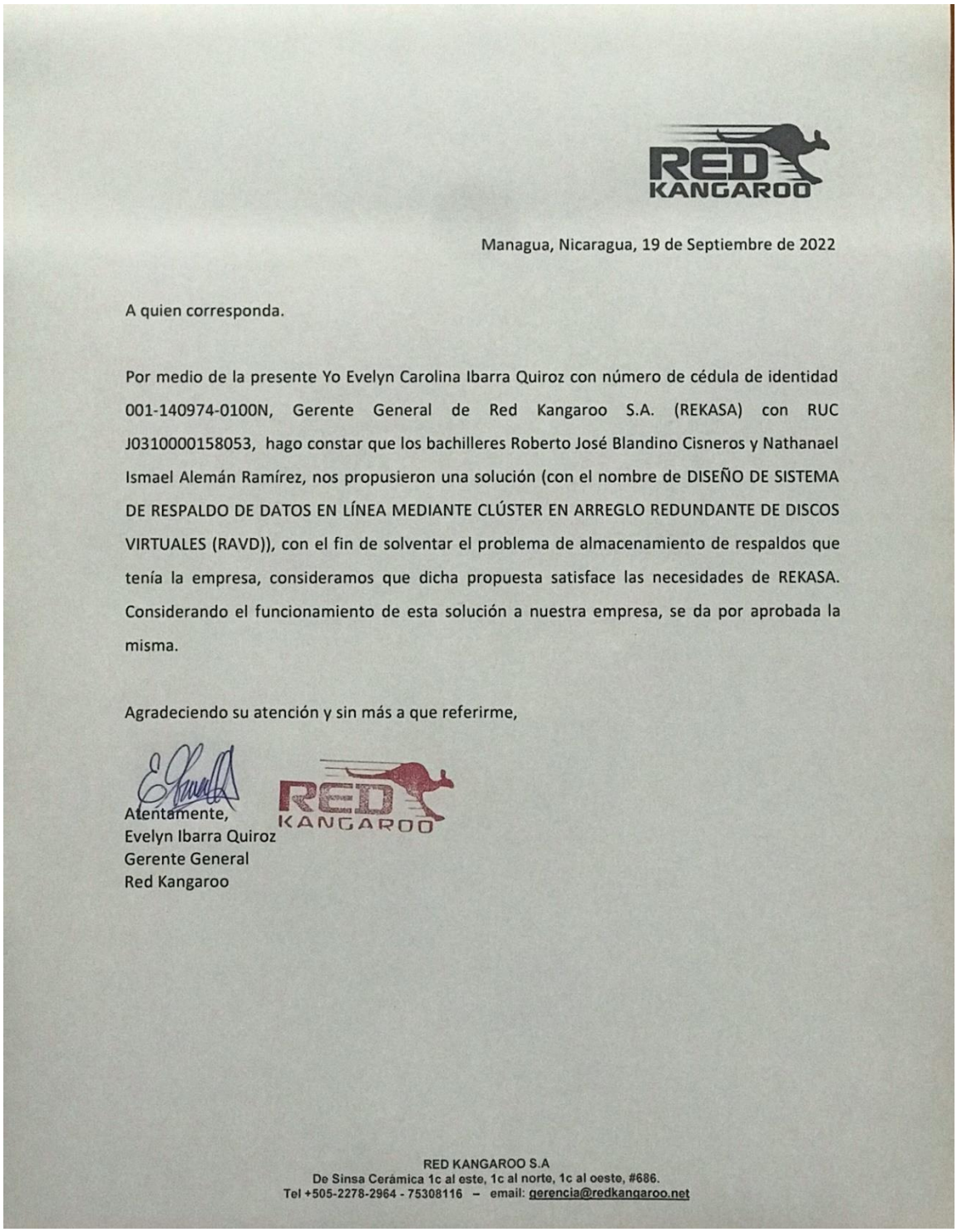
```
#!/bin/bash
time=20

function stop_cluster(){
  echo "Stopping services in nbdc1"
  systemctl stop heartbeat
  case "$(hostname)" in
    nbdc1)
```

```
      echo "Shutting down nbdc2"
      ssh root@nbdc2 "shutdown -h now >/dev/null &"
      ;;
    nbdc2)
      echo "Shutting down nbdc1"
      ssh root@nbdc1 "shutdown -h now >/dev/null &"
      ;;
  esac
  sleep $time
  for (( i=2;i<=12;i+=2 ));do
    echo "Shutting down nbds$i"
    ssh root@nbds$i "shutdown -h now >/dev/null &"
    sleep $time
  done
  for (( i=1;i<=12;i+=2 ));do
    echo "Shutting down nbds$i"
    ssh root@nbds$i "shutdown -h now >/dev/null &"
    sleep $time
  done
  case "$(hostname)" in
    nbdc1)
      echo "Shutting down nbdc1"
      ;;
    nbdc2)
      echo "Shutting down nbdc2"
      ;;
  esac
  shutdown -h now
}

case "$1" in
        stop_cluster)
                stop_cluster
                ;;
        *)
                Echo "$0 stop_cluster"
                ;;
esac
```

APÉNDICE K. Carta de aprobación de REKASA para el diseño propuesto



Managua, Nicaragua, 19 de Septiembre de 2022

A quien corresponda.

Por medio de la presente Yo Evelyn Carolina Ibarra Quiroz con número de cédula de identidad 001-140974-0100N, Gerente General de Red Kangaroo S.A. (REKASA) con RUC J0310000158053, hago constar que los bachilleres Roberto José Blandino Cisneros y Nathanael Ismael Alemán Ramírez, nos propusieron una solución (con el nombre de DISEÑO DE SISTEMA DE RESPALDO DE DATOS EN LÍNEA MEDIANTE CLÚSTER EN ARREGLO REDUNDANTE DE DISCOS VIRTUALES (RAVD)), con el fin de solventar el problema de almacenamiento de respaldos que tenía la empresa, consideramos que dicha propuesta satisface las necesidades de REKASA. Considerando el funcionamiento de esta solución a nuestra empresa, se da por aprobada la misma.

Agradeciendo su atención y sin más a que referirme,

Atentamente,
Evelyn Ibarra Quiroz
Gerente General
Red Kangaroo

RED KANGAROO S.A
De Sinsa Cerámica 1c al este, 1c al norte, 1c al oeste, #686.
Tel +505-2278-2964 - 75308116 – email: gerencia@redkangaroo.net