Universidad
Carlos III de Madrid

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA EN INFORMÁTICA

Especialidad en aplicaciones y sistemas distribuidos.

Sistema de almacenamiento secundario a medida para aulas GNU/Linux

Alumno: Roberto Andradas Izquierdo

Tutor: Alejandro Calderón Mateos

Leganés, julio de 2013.

Título:

Autor:

Director:

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día ___ de _____ de 20___ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de _____

VOCAL

SECRETARIO PRESIDENTE

AGRADECIMIENTOS

Este proyecto pone la guinda final a todo el trabajo realizado durante años para lograr el sueño de ser ingeniero. Las circunstancias han sido determinantes. Por ese motivo, quiero agradecer y recordar a todas las personas que me han apoyado.

En primerísimo lugar a mis padres, por todo su apoyo incondicional en esta aventura. Ellos me han dado un entorno adecuado para estudiar e interesarme por aprender lo que me gusta, además de enseñarme el valor del esfuerzo que ha sido la pieza clave.

A mi hermana Sandra y mi cuñado Ángel, me han ayudado con sabias palabras cuando ha sido necesario y me han hecho tío trayendo al mejor sobrino del mundo, Iván, a la familia.

A mi abuela Patro, que siempre se preocupa por la familia, mis primos Carlos y Eduardo con los que he pasado muy buenos momentos en la montaña. Y, sin dudarlo ni un segundo, también a los que ya no están. Todos ellos han construido los cimientos sobre los que me he apoyado para crecer.

Los amigos de toda la vida, Felipe, Miguel, Paco junto con Yurema y Mónica, todos son irrepetibles e insustituibles. Al señor Valiente por su gran corazón, Cajonero por su guía-cheli y su generosidad, Isabel cuyas palabras siempre me enseñan tanto, Fany que me escucha con paciencia infinita y KCHA que me pule al Quake 3 y siempre me guía.

Por supuesto también, a mis amigos de la UC3M, Luis, Javi y Álex que tan buen ejemplo a seguir son. No puedo evitar hacer una mención destacada a dos personas adorables que se han ganado lo mejor de mí, Vero que apareció en el momento oportuno y siempre sabe qué hacer, y Cris, que ha conseguido que me ilusione de nuevo.

A GSyC/LibreSoft, que me ha ayudado a compatibilizar estudios con trabajo. En especial a Luis Cañas y su optimismo, Pedro García que tuvo razón y "todo ha ido bien", Santiago Dueñas, el mismo que viste y calza, Alicia por ser encantadora, Daniel y su voz, Pedro Coca porque siempre ha estado pendiente de esto y a Felipe "the undertaker" Ortega y Jesús por creer siempre en mí.

Y por último, pero no por ello menos, a mi tutor Alejandro Calderón, siempre dispuesto a ayudarme para alcanzar la meta final.

RESUMEN

A lo largo de los últimos años, la evolución de la electrónica y las tecnologías de comunicación ha permitido plantear escenarios anteriormente impensables, como por ejemplo, la disponibilidad de la información independientemente de su ubicación. También, gracias a la mejora en las tecnologías de almacenamiento de datos, ha sido viable el almacenamiento masivo de información en cantidades nunca antes vistas.

En concreto, la aplicación de estas tecnologías en los entornos de aulas informáticas universitarias es un buen ejemplo de un entorno específico, ya que abarca diferentes aspectos como protocolos de comunicación, sistemas de ficheros, gestión de dispositivos de bloques, tolerancia a fallos, seguridad, disponibilidad, etc.

Este proyecto afronta el problema de dotar de almacenamiento a aulas de terminales GNU/Linux en un entorno universitario. Consiste en, utilizando información disponible a priori sobre sistemas similares, tecnologías actuales de almacenamiento e informes de rendimiento de las mismas, crear un sistema a medida y fácil de integrar.

El diseño del sistema pretende ofrecer una solución sencilla, optimizada, modificable, reproducible y bien documentada. Está compuesto por capas muy desacopladas que se comunican entre ellas, lo que permite que sea fácilmente ampliable en caso de que aumenten las necesidades.

ABSTRACT

Along the last years, the evolution of the electronic and communication technologies have allowed unbelievable scenarios, for example, the information availability regardless the location of it. Moreover, massive data storage never seen before has been possible through the data storage technologies improvement.

Specifically, the application of these technologies to university informatics rooms is a good case of a specific environment, it covers different aspects like communications protocols, file systems, block device management, failure tolerance, security, etc.

This project faces the problem of providing data storage to GNU/Linux terminals in a university environment. It is about, using already known information about similar systems, current storage technologies and performance reports of them, to create a tailored and easy to integrate system.

The design of the system intends to offer a simple, optimized, modifiable, reproducible and well documented solution. It is composed by several decoupled layers which communicate between them allowing an easy expanding as the necessities increase.

Índice de contenido

Índice de figuras

Índice de tablas

Capítulo 1

1. Introducción

A lo largo de la historia de los sistemas computacionales y como consecuencia directa del diseño lógico de un computador planteado por el matemático Von Neumann [1], toda un área de conocimiento entorno al almacenamiento de información ha evolucionado hasta nuestros años. Siguiendo dicho modelo, se ha innovado y creado tecnología capaz de almacenar información de forma que se garantiza su integridad y disponibilidad.

El presente proyecto muestra una solución al problema concreto del almacenamiento secundario de información, específicamente en sistemas de aulas de terminales GNU/Linux en el contexto universitario.

Las principales tareas realizadas en el proyecto tienen relación con el estudio de arquitecturas de almacenamiento, tecnologías disponibles relacionadas y la integración de diferentes componentes para crear una solución a medida fácilmente integrable.

Además, se ha considerado, como punto importante, abordar el problema en el terreno del software libre [2]. De este modo, se pretende ofrecer una solución basada en estándares, adaptable a otras situaciones similares y de bajo coste.

El proyecto ha sido realizado tratando de establecer sinergias con otros grupos de profesionales del sector, así como con usuarios de aulas GNU/Linux.

Las aulas GNU/Linux administradas por el departamento GSyC [3] de la Universidad Rey Juan Carlos [4] sirvieron como caso de estudio, a partir del cual se han tomado datos reales que han permitido crear un diseño adaptado al problema concreto.

También, se ha trabajado con el grupo de investigación GSyC/LibreSoft [5] de la misma universidad, liderado por Jesús M. González Barahona. Dicho grupo y la URJC han proporcionado el lugar físico y los recursos hardware para poder hacer realidad la solución final.

Finalmente, los resultados del análisis de este proyecto y la solución propuesta han sido ofrecidos a GSyC como agradecimiento a su ayuda, también, como posible alternativa a su sistema actual.

1.1. Motivación y objetivos

Es habitual encontrar instalaciones GNU/Linux como soluciones de almacenamiento al uso. Dichos sistemas, aunque funcionales, en ocasiones carecen a priori de un diseño optimizado que normalmente va llegando con el tiempo, causando problemas por el camino, lo que dificulta enormemente otros aspectos como la usabilidad y además mina la confianza de los usuarios en el sistema.

Por este motivo, el objetivo principal del presente proyecto es la realización de una solución de almacenamiento secundario a medida para aulas GNU/Linux, además de ser integrable en un entorno de aulas existente.

Para alcanzar este objetivo principal es necesario lograr una serie de objetivos secundarios previamente:

- Comprensión de las necesidades y características, en lo referente a almacenamiento secundario, de los terminales GNU/Linux.
- Estudio e investigación de las tecnologías existentes disponibles, en el contexto del software libre, para la creación de sistemas de almacenamiento secundario.
- Adquirir los conocimientos necesarios para el manejo de los componentes hardware y software.
- Conocer cómo afectan al rendimiento diferentes factores del manejo de datos en el sistema de almacenamiento.

1.2. Fases del proyecto

En este proyecto han existido cinco fases de desarrollo diferentes que se explican a continuación.

La primera fase correspondió a la adquisición de conocimiento realizada sobre almacenamiento secundario en sistemas computacionales. La actividad principal en esta fase ha sido la lectura de documentación relacionada con arquitecturas y tecnologías ya existentes en dicho campo. Es un campo muy amplio, por lo que la adquisición de conocimiento ha requerido abarcar diferentes niveles de abstracción. Como se verá

durante el presente documento, es necesario comprender el funcionamiento de los diferentes niveles de almacenamiento para proceder a un diseño adecuado.

La segunda fase entra de lleno en el proceso de análisis, en el cual se establece el trabajo a realizar. En dicha fase se han obtenido requisitos y casos de uso. De este modo se tiene una visión clara y concreta de qué hay que hacer, sin descuidar ningún aspecto del proyecto.

La tercera fase ha consistido en la creación de un estudio del sistema actual de aulas de la URJC y un estudio de rendimiento. El segundo estudio ha permitido determinar el comportamiento de varias tecnologías involucradas y la adquisición de conocimiento técnico. Dicho proceso de adquisición ha facilitado la configuración final del sistema y su posterior integración con un entorno de aulas ya existente.

La cuarta fase, o fase de diseño y configuración de la arquitectura, ha consistido en detallar el sistema desde el punto de vista arquitectónico y generar los procedimientos para aplicar dicho diseño. Debido a las características del sistema, se muestra un diseño único, en vez de un diseño por cada componente. En cambio, este diseño se presenta organizado por capas, atendiendo a los diferentes niveles que se necesitan para acceder o depositar la información.

La quinta fase ha consistido en la integración del sistema diseñado en un entorno de aulas GNU/Linux y en la creación y comprobación de las pruebas. Dichas pruebas validan y verifican el sistema. Al no ser un proyecto de desarrollo de software no existe fase de implementación.

1.3. Medios empleados

En esta sección se presentan los recursos hardware y software utilizados para la realización de este proyecto.

1.3.1. Recursos hardware

Los recursos hardware utilizados provienen tanto de medios personales como de medios prestados por el grupo de investigación GSyC/Libresoft de la URJC.

- 14 discos Seagate Barracuda ES.2 SATA II - ST3500320NS:
  - Tamaño de sector/bloque físico: 512 bytes
  - Velocidad de giro 7200 rpm

Ilustración 1: Seagate Barracuda ES.2.

- Controladora RAID Adaptec 31605 RAID Controller:

Ilustración 2: Adaptec 31605 RAID Controller

- Servidor de almacenamiento Supermicro 3U:
  - Intel® Core™2 Duo CPU E8200 @ 2.66GHz
  - 8 GB DDR2 RAM
  - Gigabit Ethernet interface
  - Doble fuente de alimentación 800W PWS-801-1R
  - 16 bahías de disco SATA/SAS

Ilustración 3: Supermicro 3U.

- Armario:

Ilustración 4: armario.

- Equipo de red:

Ilustración 5: equipo de red.

- Equipo de sobremesa
  - Asus motherboard
  - Intel i3
  - 4 GB DDR3 RAM
  - 250 GB Seagate HD

1.3.2.Recursos software

Los recursos software utilizados se detallan a continuación.

- **Ubuntu Server 10.04 LTS**: es una distribución de GNU/Linux para servidores creada por la empresa Canónical. Dispone de un sistema de gestión de paquetes de software muy potente y al mismo tiempo proporciona soporte durante un periodo de tiempo de 5 años.

    - **Herramientas GNU**: conjunto de herramientas creadas por GNU habituales en todas las distribuciones GNU/Linux y que permiten realizar múltiples tareas como editar ficheros, gestionar recursos y realizar cambios en el propio sistema.
    - **Linux kernel 2.6.32**: versión incorporada del *kernel* Linux por defecto en la distribución Ubuntu Server 10.04 LTS.

Ilustración 6: logotipo de Ubuntu.

- **IOzone 3.308**: es un software de *benchmark* para la entrada y salida de datos sobre un sistema de ficheros.

- **The R Project for Statistical Computing**: es un paquete de software para estadística, es muy potente y ejecuta en varias plataformas como Windows y GNU/Linux.

Ilustración 7: logotipo de R.

- **OpenSSH client**: cliente que permite conectarse a servidores SSH.

Ilustración 8: logotipo de OpenSSH.

- **GNU Emacs 23.1.1**: editor de textos del proyecto GNU con el cual se ha trabajado en la configuración de servicios y desarrollo de scripts.

Ilustración 9: logotipo de GNU Emacs.

- **Microsoft Office 2010**: suite ofimática para la creación de documentos y presentaciones. Con este software se ha creado el presente documento y la presentación del proyecto.
  - Microsoft Excel 2010
  - Microsoft PowerPoint 2010
  - Microsoft Word 2010
  - Microsoft Project 2010

Ilustración 10: logotipo de Microsoft Office 2010.

- **Windows 7**: sistema operativo sobre el cual se ha trabajado durante la utilización del anterior producto software.

Ilustración 11: logotipo de Windows 7.

1.4. Estructura del documento

En esta sección se presenta la estructura del documento, dando una descripción general del contenido de cada capítulo.

- **Capítulo 1 – Introducción**: presenta motivación y objetivos, fases del proyecto, medios empleados y la estructura del documento.

- **Capítulo 2 – Estado del arte**: presenta arquitecturas y tecnologías ya existentes en el campo del almacenamiento.

- **Capítulo 3 – Análisis**: muestra una serie de requisitos y casos de uso que recogen, de forma organizada, las necesidades que ha tenido que satisfacer el proyecto realizado.

- **Capítulo 4 – Estudio del sistema actual y tecnologías relacionadas**: explica los estudios, previos al diseño, realizados sobre el sistema actual de almacenamiento de las aulas de la URJC y las tecnologías relacionadas con almacenamiento secundario.

- **Capítulo 5 – Diseño y configuración**: expone el diseño del sistema creado y la configuración de sus componentes.

- **Capítulo 6 – Integración y pruebas**: describe el proceso de integración del sistema en un entorno de aulas GNU/Linux y las pruebas que validan y verifican el sistema.

- **Capítulo 7 – Planificación y presupuesto**: recoge la planificación seguida durante el proyecto y el presupuesto del mismo.

- **Capítulo 8 – Conclusiones y trabajos futuros**: último capítulo dónde se elaboran las conclusiones y se proponen las posibles líneas de trabajos futuros a realizar, tomando este proyecto como base.

- **Anexo A – Resultados de la encuesta de integración**: resultados de la encuesta realizada a profesionales sobre la facilidad de integración.

- **Anexo B – Procedimientos**: procedimientos creados y utilizados para la creación e integración de la solución.

- **Glosario de términos**: incluye un listado de términos y su descripción, para una mejor comprensión del presente documento.

- **Bibliografía**: muestra las referencias utilizadas en la elaboración del proyecto, sobre las cuales se basa gran parte del trabajo realizado.

Capítulo 2

2. Estado del arte

Este capítulo recoge la primera fase del proyecto, muestra una visión de las arquitecturas y tecnologías ya existente en el campo del almacenamiento.

2.1. Introducción

Debido a la naturaleza del proyecto, dónde gran parte del trabajo ha consistido en crear un sistema de almacenamiento, es necesario presentar, desde un punto de vista lógico y arquitectural, el lugar dónde el proyecto encaja dentro de un sistema de computación. Para ello se presenta, a continuación, el modelo lógico vigente en la mayoría de sistemas en producción.

A mediados de la década de los cuarenta, el científico matemático, de procedencia húngara, Von Neumann publicó el artículo "*First Draft of a Report on the EDVAC*" [1]. Este artículo puso las bases de un modelo lógico de sistema de computación automática.

En dicho artículo, realiza distintas definiciones que describen el comportamiento general del dispositivo. De todas ellas, es de particular interés la que indica que las instrucciones que gobiernan cada operación a realizar por el sistema, deben ser proporcionadas al dispositivo de forma completamente detallada. Dichas instrucciones incluyen toda la información numérica necesaria para resolver el problema. Estas instrucciones deben ser proporcionadas al dispositivo de forma que puedan ser leídas por él mismo.

Algunos de los medios sugeridos por Von Neumann para proporcionar dichas instrucciones y datos al sistema son:

- Tarjetas perforadas
- Cinta de teletipo
- Instrucciones magnéticamente impresas en acero.

- Instrucciones fotográficamente impresas en cinta de película.

Además, el sistema, tras recibir las instrucciones y ejecutarlas sin necesidad de la intervención de un humano, generará unos resultados que deben ser escritos de alguna de las formas descritas anteriormente.

En el artículo se indica que se asume que el sistema funcionará sin fallos. En cambio, admite que el uso de cualquier dispositivo trae como consecuencia la posibilidad finita de fallos, que podrían provocar falsos resultados. Por este motivo, se aconseja la intervención de inteligencia humana para resolver los problemas que pudieran surgir durante el uso del dispositivo. A pesar de todo, se explica que sería posible crear un sistema capaz de detectar la mayoría de problemas, indicándolos al exterior mediante el uso de algunas señales para posteriormente realizar una parada. Bajo ciertas circunstancias, podría ser el propio sistema el que solucionase el problema de forma automática y continuase con la operación.

Además, Von Neumann establece una serie de subdivisiones lógicas en el sistema, las cuales identifica mediante siglas. A continuación, se muestran las distintas subdivisiones con una breve definición de cada una de ellas:

- **Central aritmética (CA)**: realiza operaciones aritmética elementales como suma, resta, multiplicación y división.

- **Central de control (CC)**: realiza la secuenciación de las operaciones de forma eficiente.

- **Memoria (M)**: operaciones complejas necesitan de gran cantidad de memoria para almacenar las instrucciones y los valores numéricos que la forman.

- **Soporte de grabación exterior (R)**: el soporte en el cual se proporcionan las instrucciones, los datos al sistema y también es el soporte dónde se escriben los resultados de las operaciones.

- **Input (I)**: es el subsistema que lee los valores de R y los introduce en M.

- **Output (O)**: es el subsistema que lee los valores resultantes de las operaciones de M y los escriben como resultados en R.

Von Neumann especifica en el artículo la necesidad de que las comunicaciones de entrada y salida con R siempre sean mediante M, nunca directamente con el subsistema CC. El motivo es la mejora considerable de rendimiento que se produce al permitir que CC trabaje directamente con M, en vez de con el original, R, de tarjeta perforada o cinta de teletipo.

A continuación, se presenta un diagrama con la división en subsistemas de un sistema de computación automática siguiendo el modelo lógico de Von Neumann.

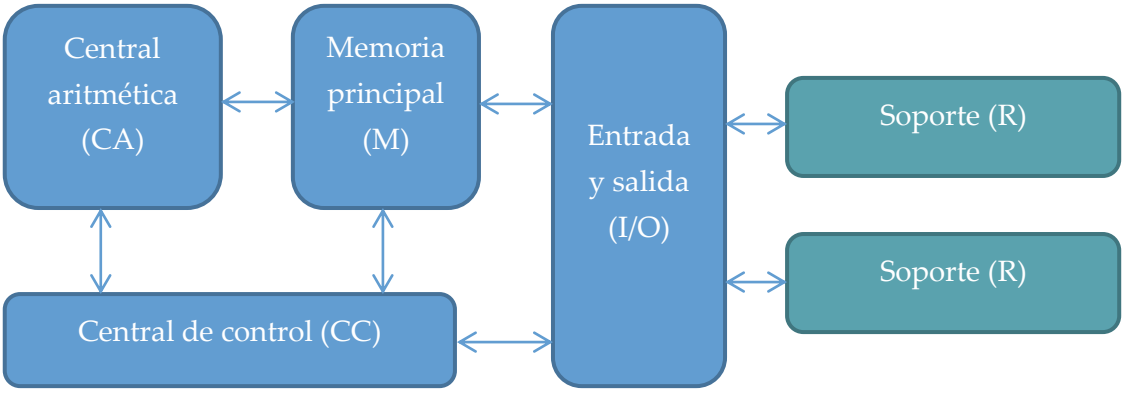

Ilustración 12: modelo lógico de Von Neumann.

Se debe resaltar, que el modelo presentado por Von Neumann es un modelo muy general. Las implementaciones de los sistemas actuales incorporan otros subsistemas que añaden funcionalidad extra, tanto en forma de componentes hardware como en forma de capas superiores de abstracción, las cuales han permitido llegar a sistemas computacionales como los actuales.

En el contexto del proyecto, los terminales de las aulas GNU/Linux son una implementación de este modelo lógico. Concretamente, la zona del modelo de Von Neumann que abarca el trabajo hecho en este proyecto desde el punto de vista de los terminales es el subsistema R o también conocido como sistema de almacenamiento masivo en los sistemas actuales. En cambio, el proyecto propone una solución que abarca también aspectos de los subsistemas I/O y M.

2.1.1.E/S y almacenamiento

Al igual que en el modelo presentado por Von Neumann, los computadores actuales no se conciben sin un sistema de entrada y salida (E/S a partir de ahora). La unidad de proceso central de un computador, CA + CC, no servirá de nada sin dispositivos que se encarguen de almacenar los datos, los programas y que permitan interactuar a los usuarios.

Los dispositivos encargados del almacenamiento de datos y programas se denominan dispositivos de almacenamiento secundario (discos) y terciario (cintas y sistemas de archivo). Proporcionan almacenamiento no volátil de datos. Su función principal es abastecer de datos y almacenamiento a los programas que se ejecutan en la unidad central de proceso del computador. Dependiendo de la capacidad y la celeridad con la que estos dispositivos proporcionan los datos, se pueden dividir en almacenamiento secundario o terciario [6], como se observa en la siguiente ilustración.

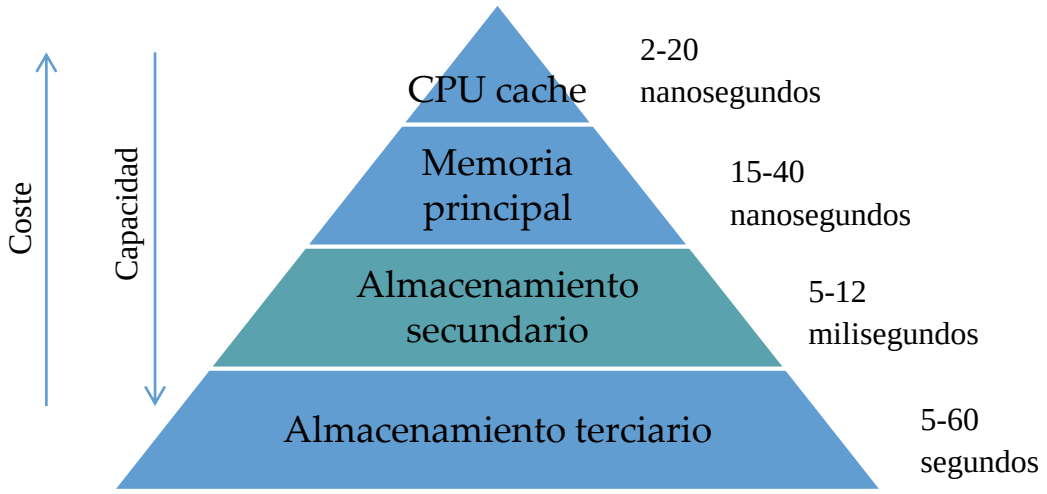

Ilustración 13: jerarquía de dispositivos de E/S. [7] [8] [9] [10]

Los dispositivos de interactuación con el usuario también se consideran dispositivos de E/S pero no se profundizará en ellos durante este documento por salirse del ámbito del proyecto.

Además, existen otro tipo de dispositivos de E/S que son utilizados para conectar el computador con otros computadores a través de una red. Principalmente se utilizan módems, para comunicación mediante la red telefónica, y tarjetas de interfaz de red para conectar el computador a una red de área local.

2.1.1.1. *Almacenamiento secundario*

En concreto, la capa de almacenamiento secundario se utiliza para guardar los programas y datos en dispositivos razonablemente rápidos. A esta información, se accede mediante el sistema de ficheros.

Los discos son dispositivos físicos que contienen la información. Tienen determinadas características que influyen en su funcionamiento y rendimiento. Son dispositivos en su gran mayoría electromecánicos. Durante este proyecto únicamente se utilizarán discos duros. Otro tipo de dispositivos como discos ópticos, extraíbles o basados en memorias flash, como los modernos SSD, no han sido considerados por falta de recursos hardware.

A continuación, se muestra la geometría física habitual de los discos duros. Dicha geometría es importante porque de su diseño depende la capacidad del disco y gran parte del tiempo que el mismo tarda en llegar a la zona dónde se desea escribir o leer.

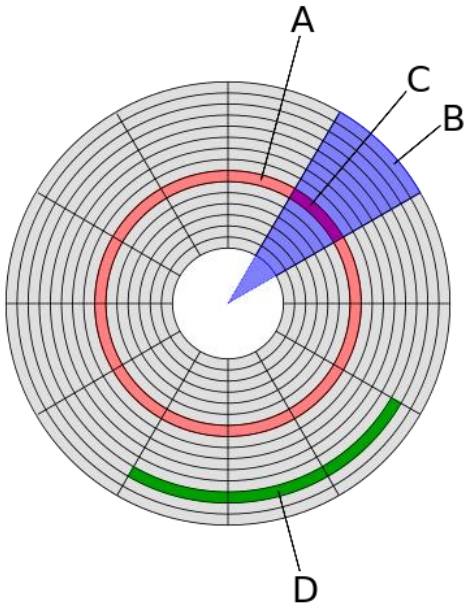

Ilustración 14: geometría física de disco duro. [11]

- **A – Pista**: es una zona circular de la superficie del disco en el cual se almacena o se lee la información. Cada pista tiene distinta densidad o número de sectores, lo que aumenta la complejidad del acceso [12].

- **B – Sector geométrico**: define un área de la superficie del disco que corresponde al ángulo que forman dos radios del disco que pasan por los dos bordes de un mismo sector.

- **C – Sector/Bloque**: es la subdivisión de una pista. En él, se almacena o se lee una cantidad fija de información. Tradicionalmente, según el modelo definido por IBM en 1970 y llamado FBA [13] [14] [15], los sectores tienen un tamaño de 512 bytes. Los discos más modernos utilizan sectores de 4 KB para poder alcanzar las capacidades que se ven hoy en día, ajustándose al estándar *Advanced Format* [16] definido por IDEMA [17].

- **D – Cluster**: es un conjunto de sectores consecutivos. Por motivos de eficiencia, en la escritura o lectura de datos contiguos, se intenta utilizar sectores consecutivos.

En realidad, un disco duro está formado por una cantidad finita de varios discos, como el mostrado en la figura anterior, apilados, de forma que las cabezas lectoras se sitúan en el espacio intermedio. Al conjunto de pistas de cada disco, que coinciden en la misma vertical, se le denomina cilindro por la forma tridimensional que forman.

Estos parámetros, junto con otros como la velocidad de rotación del disco, tiempo de latencia (tiempo que tarda en llegar el dato que está bajo la cabeza lectora) y tiempo de búsqueda, tiempo en ir de una pista a otra. Afectan de forma drástica al rendimiento del sistema, a la hora de escribir o leer datos, en su capa de almacenamiento secundario.

Por último, para terminar con la descripción de disco, se debe mencionar la estructura lógica que presenta. Dicha estructura lógica es la manejada por el sistema operativo del computador y facilita el acceso a los datos almacenados en el disco, por lo que no es necesario tener conocimiento de la geometría física del mismo.

Esta estructura lógica está formada por una serie de particiones o agrupaciones de sectores, que conforman una partición, y una tabla que sirve de índice, el cual indica la localización de cada una de las particiones. Las arquitecturas x86 y x64 requieren de esta estructura lógica para poder proceder al inicio del sistema operativo. Esto permite dividir el espacio proporcionado por un disco como si fueran varios discos distintos y, por lo tanto, realizar una gestión independiente de cada partición.

Existen dos métodos de tabla de particiones de discos, MBR y GPT. Ambos hacen referencia a cómo se organiza el índice que referencia las diferentes particiones. Se presentan a continuación.

MBR fue desarrollado durante los años ochenta, es el más extendido incluso hoy en día. Este modelo tiene algunas desventajas importantes, como la limitación de 2TB en el tamaño de las particiones que se pueden crear o la facilidad con que la tabla de particiones se puede corromper.

A continuación, un plano de particionamiento utilizando MBR:

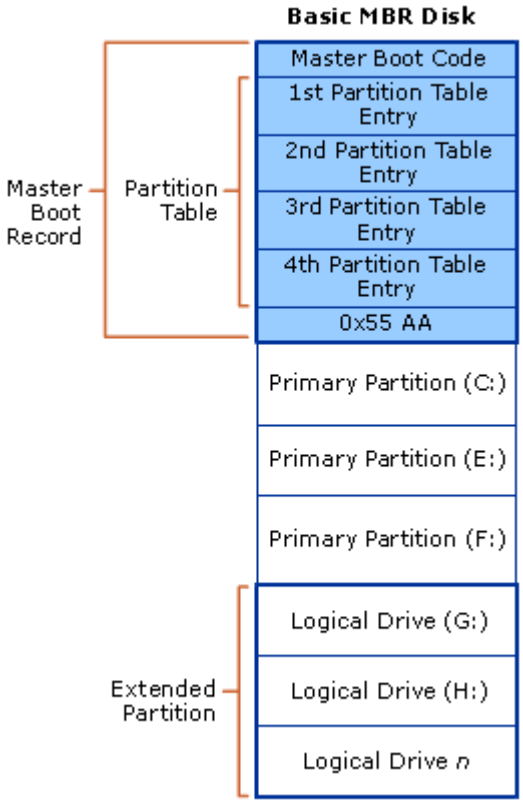

Ilustración 15: particionamiento con MBR. [18]

La figura muestra una primera zona en azul que representa el MBR, esta zona, cuyo inicio se marca con el *Master Boot Code* y cuyo fin con el valor "0x55 AA", contiene la tabla de particiones. La tabla de particiones es un listado con las entradas que hacen referencia a cada partición del disco.

El MBR ocupa exactamente 512 bytes. Esto es así porque se diseñó en una época en la que el sector de los discos era de 512 bytes, por este motivo el MBR siempre se coloca en el primer sector de un disco.

GPT fue desarrollado a finales de los años noventa como respuesta a los problemas presentados por MBR en la creciente industria del almacenamiento. Además, ofrece capacidad de crear un mayor número de particiones, hasta 128, así como la posibilidad de crear particiones de mayor tamaño. Además, aporta mayor seguridad ante la corrupción de la tabla de particiones mediante el uso CRC.

A continuación, un plano de particionamiento utilizando GPT:

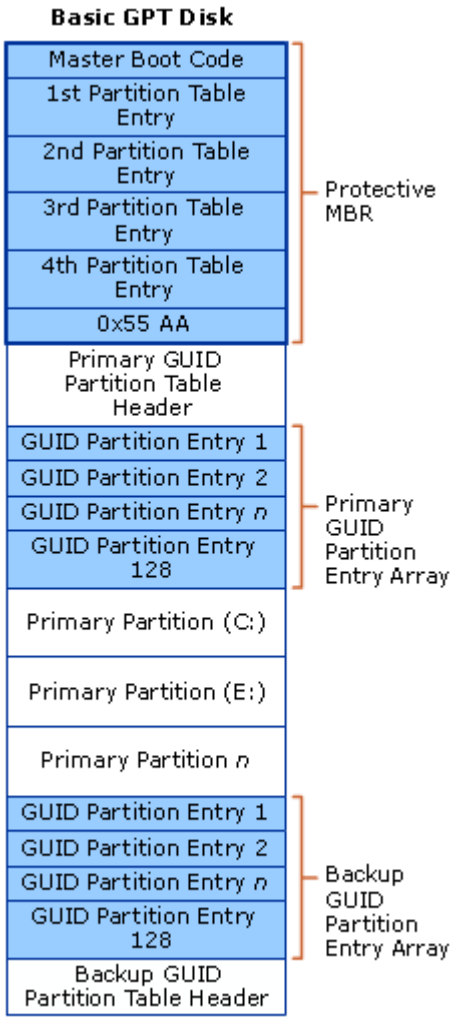

Ilustración 16: particionamiento con GPT. [18]

La figura muestra una primera zona inicial idéntica al MBR, la cual sirve para proporcionar compatibilidad hacia atrás. Un disco particionado con GPT necesita más

de 512 bytes para almacenar su información de particionado, y por ello, más de un sector en un disco con tamaño de sector de 512 bytes.

A continuación, se incluye la cabecera *Primary GUID Partition Table Header* que indica el número de sectores que pueden ser utilizados para almacenar entradas de particiones, también incluye un CRC para comprobar la integridad de la propia cabecera.

Después, se puede observar la tabla de particiones o *Primary GUID Partition Entry Array*, si usamos la terminología introducida por GPT. En esta tabla, cada entrada contiene un identificador global único que identifica el tipo de dato almacenado en la partición, cómo la partición es utilizada y la posición de la partición.

Por último, después de todas las particiones, se observa una copia de seguridad de la *GUID Partition Entry*.

2.1.1.2. Dispositivos de bloque

Un dispositivo de bloque es un dispositivo, por ejemplo un disco duro, que alberga datos. Además, soportan operaciones de E/S que transfieren uno o más bloques. Dichos bloques tienen un tamaño de 512 bytes habitualmente, ya que corresponde al tamaño del sector de un disco duro. Como se dijo en la sección 2.1.1.1, este tamaño puede ser de 4 KB en discos duros modernos.

En el sistema operativo GNU/Linux, los dispositivos de bloques son accedidos mediante el propio sistema de ficheros. El sistema operativo ofrece un interfaz en el directorio */dev/* para comunicar con dichos dispositivos. Debido a la abstracción del sistema operativo, la comunicación con los dispositivos de bloques se realiza leyendo o escribiendo cadenas de bytes, sin necesidad de tener en cuenta que dichas operaciones deben ser traducidas a operaciones de E/S de uno o más bloques.

Es decir, es posible escribir o leer bytes en el fichero */dev/sda1*. Dicho fichero representa un dispositivo de bloques. Habitualmente, el sistema proporciona un fichero en */dev/* por cada partición existente en el disco.

Esta es la forma en la que los dispositivos de almacenamiento secundario son accedidos realmente por un sistema GNU/Linux en sus capas inferiores.

2.1.2. Arquitecturas de almacenamiento

El almacenamiento es, por lo tanto, una parte importante de un sistema computacional. Sin él, los sistemas actuales serían inconcebibles.

Las necesidades de un sistema casero no son las de un servidor en una universidad o en una gran corporación. Por este motivo, la forma en la que se dota a un sistema computacional de almacenamiento secundario ha evolucionado, generando un área de negocio en la que, hasta la actualidad, existen varias arquitecturas dependiendo de cómo se conectan los dispositivos de disco.

A continuación, se muestra una definición resumida de cada arquitectura:

- **DAS** (*Direct Attached Storage*): hace referencia a dispositivos de discos conectados directamente en el computador mediante protocolos ATA, SATA, eSATA, SCSI, SAS o fibra.

- **NAS** (*Network Attached Storage*): hace referencia a un sistema computacional que proporciona almacenamiento a nivel de sistema de ficheros a varios computadores clientes, los cuales pertenecen a una red de datos.

- **SAN** (*Storage Area Network*): hace referencia a un sistema computacional que proporciona almacenamiento a nivel de dispositivo de bloque a varios computadores clientes, los cuales pertenecen a una red de datos.

- **Cloud**: hace referencia a un modelo en el que el almacenamiento se proporciona de forma elástica en función de las necesidades y mediante servicios online.

2.1.2.1. DAS

En la arquitectura DAS los dispositivos de disco son conectados directamente al computador que los va a utilizar, ya sea internamente mediante ATA, SATA, SCSI y SAS o externamente mediante eSATA o USB.

DAS es una consecuencia de los primeros diseños de los antiguos computadores dónde el concepto de red de computadores no estaba muy extendido, por lo que debían ser autónomos y esto requería almacenamiento secundario en el propio sistema.

A pesar de proporcionar almacenamiento secundario a velocidades relativamente buenas, DAS tiene algunos inconvenientes que lo hacen poco propicio en entornos dónde los requisitos, en cuanto a almacenamiento, son elevados.

A continuación, algunos de los inconvenientes, relacionados con la naturaleza distribuida de los sistemas DAS.

- Es complicado relocalizar el espacio disponible en un sistema para dárselo a otro que lo necesite. Requiere intervención física y en ocasiones incluso apagar ambos sistemas, aunque los fabricantes de hardware han creado mecanismos que permiten conectar y desconectar discos sin apagar el sistema. Esto es problemático cuando no es sencillo estimar el uso de almacenamiento secundario que se hará, en sistemas a los que no es fácil acceder físicamente o aquellos que requieren disponibilidad 24h.

- Realizar copias de seguridad de sistemas distribuidos cuyo almacenamiento secundario se basa en DAS es lento y complicado. Por motivos como este, las compañías no suelen realizar copias de seguridad de sistemas basados en DAS y únicamente lo hacen de servidores, debido a la importancia estratégica en el negocio de la misma.

A continuación, una imagen de una instalación DAS basada en protocolo SATA de un computador de sobremesa típico.

Ilustración 17: conexión SATA en arquitectura DAS.

En la imagen, se aprecian tres discos duros en un chasis de computador de sobremesa, dichos discos están conectados al sistema mediante un interfaz SATA (zona clara de la imagen).

Esta arquitectura no es adecuada para el problema a resolver, debido principalmente a su naturaleza distribuida.

2.1.2.2. NAS

En la arquitectura NAS existen dispositivos de almacenamiento conectados a la red, en dicha red diferentes computadores clientes los utilizan como almacenamiento secundario.

En NAS los computadores clientes ignoran la localización real física de los discos y, únicamente, se limitan a leer o escribir de su sistema de ficheros. De este modo dejan a

las capas inferiores (controladas por módulos del *kernel* en GNU/Linux) los detalles. Detalles como, por ejemplo, que un determinado fichero o directorio está realmente en un dispositivo NAS y que es necesario utilizar la red para realizar la operación solicitada.

A continuación, un diagrama de red general de un dispositivo NAS proporcionando almacenamiento secundario a clientes en una misma red.

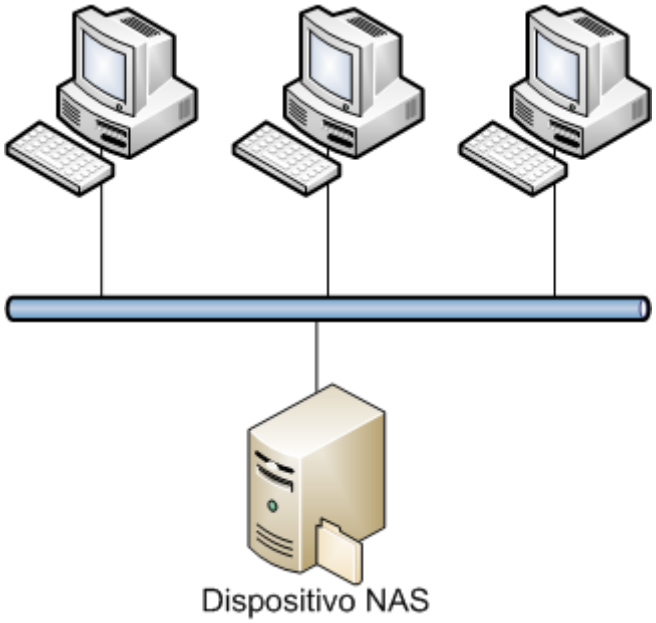

Ilustración 18: arquitectura NAS.

En esta ilustración varios clientes conectados a una red utilizan los sistemas de ficheros del dispositivo NAS. Esta arquitectura proporciona varias ventajas a destacar:

- La gestión del almacenamiento y el despliegue de esta arquitectura es sencillo.
- El coste de la implantación de una arquitectura NAS es relativamente bajo.
- Seguridad, los protocolos utilizados en arquitecturas NAS permiten aprovechar la funcionalidad de los sistemas de ficheros, por lo que existe la posibilidad de autenticación, gestión de permisos, usuarios y grupos.
- Confiabilidad, en arquitecturas NAS es sencillo realizar copias de seguridad de los datos, de forma que es posible recuperarlos tras una catástrofe.

La arquitectura NAS presenta también una serie de características que pueden derivar en problemas en algunos casos:

- Bajo rendimiento, habitualmente las arquitecturas NAS no proporcionan un hardware demasiado especializado en almacenamiento, como por ejemplo cabinas de discos de alto rendimiento ni una infraestructura de red mejor que Ethernet.

- Problemas de escalabilidad, causados por problemas de rendimiento y la dificultad para replicar dispositivos NAS manteniendo coherencia de datos.

- Homogeneidad, la arquitectura NAS dificulta la creación de un entorno heterogéneo de clientes en la red, debido a la capacidad limitada de los dispositivos NAS de proveer distintos sistemas de ficheros a los clientes.

Esta arquitectura proporciona un modelo válido para la realización del proyecto, se adapta a los recursos disponibles y sus desventajas podrían no suponer un problema en el contexto del proyecto.

2.1.2.3. SAN

Al igual que en la arquitectura anterior, existen dispositivos SAN que proporcionan almacenamiento secundario a clientes conectados a la misma red. Dichos dispositivos están especializados en tareas de almacenamiento secundario, proporcionando capacidad para gran cantidad de discos, *hot swap*, fuentes de alimentación dobles y otras características hardware que dependen de la gama y el fabricante del dispositivo.

En la arquitectura SAN, los dispositivos de almacenamiento permiten el acceso a dispositivos de bloque por parte de los clientes a través de una red de datos, dejando en manos de estos la gestión del sistema de ficheros. De este modo, un dispositivo SAN únicamente gestiona dispositivos de bloque y a su vez únicamente sabe realizar operaciones de E/S relacionadas con leer y escribir bloques, tal como se explicó en la sección 2.1.1.2.

A continuación, un diagrama de red general de un dispositivo SAN proporcionando almacenamiento secundario a clientes en una misma red.

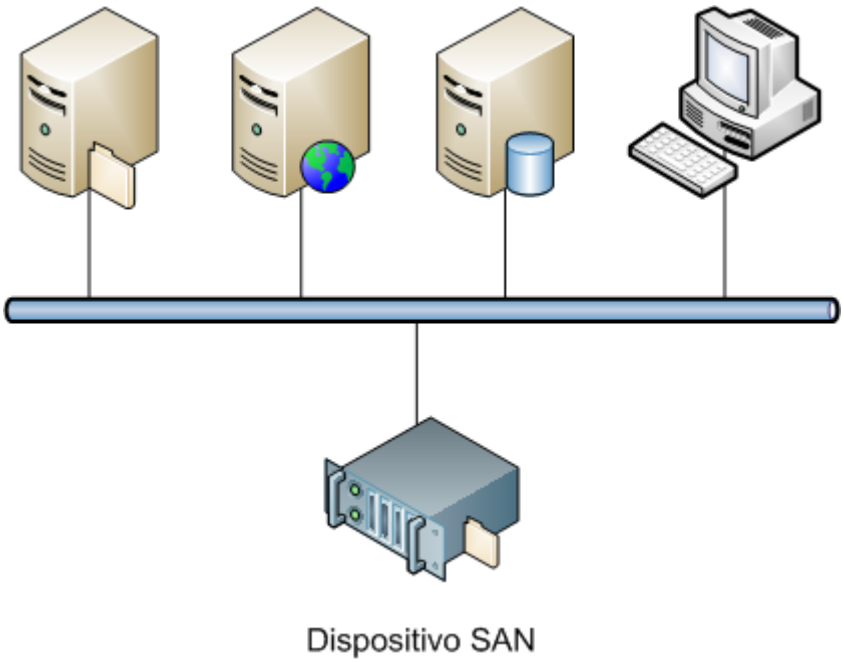

Ilustración 19: arquitectura SAN.

En el diagrama se observa un dispositivo SAN proporcionando dispositivos de bloque a varios clientes en la misma red. Es posible que los clientes sean otros dispositivos de almacenamiento, como por ejemplo dispositivos NAS, servidores web o de bases de datos o incluso computadoras de sobremesa, aunque esto último no es común debido a que complica la administración del sistema de almacenamiento en el lado del equipo de sobremesa.

Esta arquitectura proporciona las siguientes ventajas a destacar:

- Escalabilidad, es posible crear una red de dispositivos SAN dando servicio a numerosos clientes. Dicha red puede incluso estar repartida geográficamente, de forma que clientes en diferentes localizaciones dispongan de acceso a los datos que están consolidados en los dispositivos SAN de la red. Esta característica es clave en entornos de *cloud computing*.

- Confiabilidad, es muy sencillo replicar dispositivos SAN en contextos donde la alta disponibilidad es una necesidad, reduciendo la posibilidad de pérdida de datos.

- Seguridad, la estructura de la arquitectura SAN y la forma en la que se manejan los datos hacen muy difícil accesos no autorizados.

- Rendimiento, los dispositivos SAN tradicionalmente funcionan en redes especializadas de almacenamiento que tienen la característica de ser muy rápidas. Además, los dispositivos SAN, habitualmente, proporcionan hardware muy especializado, siendo capaces de albergar grandes conjuntos de discos de alto rendimiento en una sola cabina.

En cambio, la arquitectura SAN tiene una serie de características que pueden derivar en problemas en algunos casos:

- La industria SAN carece de estándares ampliamente extendidos, lo que provoca problemas de interoperabilidad entre productos de diferentes fabricantes.

- El coste de montar una arquitectura SAN es alto, desde la infraestructura necesaria hasta el personal especializado necesario. Aunque, a largo plazo y en entornos donde la escalabilidad es un requisito, esta arquitectura puede acabar siendo no tan costosa.

A pesar de que esta arquitectura podría aportar parte de la solución al problema, requiere una cantidad de recursos de los que no se dispone debido a que, únicamente, gestiona dispositivos de bloques y no sistemas de ficheros.

2.1.2.4. Cloud

En los últimos años ha tomado relevancia un nuevo paradigma de computación llamado *cloud*. Aparecen conceptos como *Infrastructure as a service* (IaaS), bajo los cuales los que los proveedores de dichos servicios ofrecen máquinas virtuales, redes, firewalls y almacenamiento.

Todos los servicios ofertados por proveedores se ofrecen, habitualmente, mediante Internet bajo el nombre de *public cloud*. En el área del almacenamiento, dichos proveedores ofrecen tanto acceso a dispositivos de bloque como a sistemas de ficheros. Existen varios proveedores de este tipo de servicios como Google Compute Engine, HP Cloud, Joyent, Rackspace o Amazon S3.

Las principales ventajas para el cliente en un modelo de almacenamiento en un *cloud* público son:

- De un modelo capex, a un modelo opex. En un modelo capex es el usuario el que gasta recursos en comprar y mantener hardware (que se devalúa rápidamente con el tiempo). En un modelo opex el usuario utiliza un servicio y paga por él en función de su uso. De este modo el cliente evita todas las complicaciones relacionadas con infraestructura.

- Escalabilidad y elasticidad, el usuario puede requerir del servicio más o menos cantidad de recursos bajo demanda. Si necesita más almacenamiento puede automáticamente extender la capacidad casi en tiempo real sin preocuparse de cómo.

- Fiabilidad, el servicio ejecuta sobre una plataforma de alta disponibilidad que permite reducir los tiempos de caída al usuario. Habitualmente los proveedores incluyen diferentes grados de fiabilidad, en función de lo requerido por el usuario.

- Uso de estándares para proporcionar el servicio, el usuario puede utilizar una gran variedad de dispositivos para acceder al servicio. No solo PCs, también teléfonos, tabletas y prácticamente cualquier dispositivo capaz de hablar TCP.

En cambio, también existe un problema muy importante, sobre todo en el uso de *cloud* para almacenamiento.

- Seguridad, es el gran enemigo del *cloud*. El uso de este modelo para almacenamiento significa la pérdida de control sobre los datos por parte del cliente, en ocasiones datos sensibles de importante valor para personas o compañías. Los sistemas de *cloud*, habitualmente, son implementados mediante grandes granjas de servidores, dónde la información de distintos usuarios no relacionados se encuentra distribuida. Esto añade incluso más dificultad a la hora de poder acceder a informes sobre qué está pasando con cierta información o dónde está porque, sencillamente, ni el proveedor lo conoce en muchas ocasiones.

Este modelo, aunque válido para recrear una solución, no es admisible en el contexto de aulas de universidad GNU/Linux dónde el control sobre los datos es importante de cara al control de las propias materias que se cursan. Probablemente, también debido a los tiempos de acceso a los datos de un servicio de *cloud* público en Internet.

2.2. Tecnologías de almacenamiento

A continuación se presentan diversas tecnologías de almacenamiento aplicables a distintos aspectos de dicho área. Todas estas tecnologías son, actualmente, usadas en entorno de producción y han sido consideradas en la solución diseñada en este proyecto.

2.2.1.RAIDs

Un RAID de discos es un conjunto redundante de discos independientes. Un RAID combina este conjunto de discos de forma que ofrece una vista lógica del mismo. Los datos son repartidos dependiendo de cómo se combinen los discos que forman el RAID.

La forma en la que se combinan los discos se denomina nivel de RAID y tiene que ver con la redundancia y el rendimiento que proporciona. Cada nivel de RAID ofrece unas características diferentes a tener en cuenta en función de las necesidades. Habitualmente se utilizan discos idénticos para la creación de RAIDs, aunque técnicamente existe la posibilidad de usar discos diferentes obligando a que el RAID creado funcione limitado por el disco de menores prestaciones.

La creación de RAIDs puede realizarse mediante la utilización de hardware especializado o mediante software. La primera opción es totalmente ajena al sistema operativo, el cual identifica el dispositivo lógico creado por el hardware como un disco físico dónde escribir o leer, aumenta generalmente el rendimiento del RAID pero dificulta el aprendizaje teniendo que tener en cuenta el funcionamiento específico de dicho hardware para el mantenimiento de los RAIDs. La segunda opción consiste en delegar al sistema operativo el control del RAID, lo cual permite gestionarlo con las herramientas habituales proporcionadas por él mismo. En cambio, esto puede ser perjudicial para el rendimiento del sistema de almacenamiento.

En esta sección se presentan los niveles de RAID más habituales. Existen multitud de niveles de RAID, los explicados en esta sección son aquellos existentes en la mayor parte de fabricantes de la industria y en el sistema operativo GNU/Linux.

2.2.1.1. RAID 0

Este nivel de RAID destaca por su alto rendimiento y capacidad adicional de almacenamiento. En cambio, cualquier fallo en los discos que lo forman causa la destrucción del *array* y por lo tanto de los datos que contiene, es decir, no ofrece tolerancia a fallos. La probabilidad de fallo aumenta con el aumento del número de

discos que forman el *array*. El cuanto a espacio disponible es la suma total del espacio ofrecido por cada disco.

A continuación, un diagrama explicativo de RAID 0:

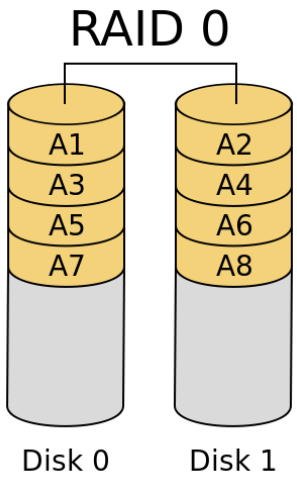

Ilustración 20: RAID 0. [19]

Consiste en la agrupación de discos de forma que el dato es dividido en *stripes*, los cuales son repartidos entre los discos de forma homogénea.

Este RAID se ha descartado debido a que no ofrece redundancia.

2.2.1.2. RAID 1

Este nivel de RAID destaca por replicar los datos de un disco en otro, creando un *mirror*, o espejado de los datos. Por lo tanto, en un RAID 1 se podría perder por completo uno de los discos sin pérdida de datos. El *array*, continuaría siendo operativo mientras quede al menos un disco en funcionamiento. Las operaciones de lectura son recibidas por los dos discos propiciando que conteste el más rápido, en cambio, durante las operaciones de escritura, no se considera que el dato está consolidado hasta que ambos discos escriben el dato, es decir, hasta que el más lento de ambos lo hace.

A continuación, un diagrama explicativo de RAID 1:

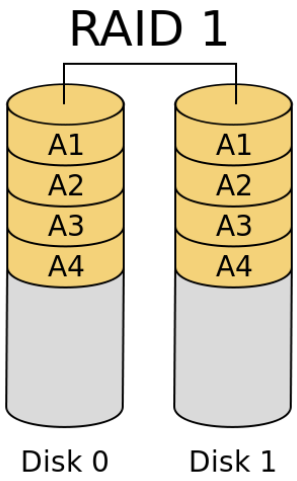

Ilustración 21: RAID 1. [20]

Se observa cómo cada dato es copiado en ambos discos, la capacidad del *array* es la mitad del espacio total resultado de la suma de la capacidad de cada disco.

Este RAID se consideró como parte de la solución debido a su redundancia. Especialmente para atender baja carga.

2.2.1.3. RAID 5

Este nivel de RAID destaca por dividir el dato en *stripes*, al igual que el nivel RAID 0, y hacer un cálculo de paridad del dato que es escrito en uno de los discos miembros de forma distribuida, es decir, no todos los cálculos de paridad se escriben en el mismo disco. El cálculo de la paridad es un hándicap de este tipo de RAID, aunque se ve compensado en ocasiones con la capacidad de realizar escrituras en paralelo de un dato como el nivel RAID 0 y además soporta el fallo de uno de los discos del *array*. Además, las lecturas se realizan en paralelo utilizando varios discos. Esto implica que añadir discos al *array* aumentará, por norma general, su rendimiento. El ratio de espacio total disponible en este nivel es $1 - (1/n)$, siendo n el número total de discos que forman el *array*.

A continuación, un diagrama explicativo de RAID 5:

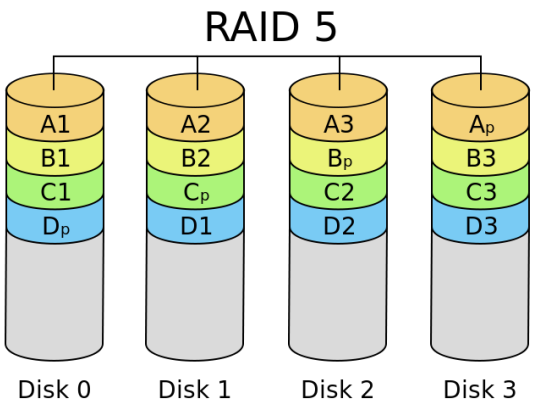

Ilustración 22: RAID 5. [21]

Se observa como el dato A es dividido en tres partes y a su vez se calcula la paridad, *Ap*, la cual se escribe en el cuarto disco. El resto de paridades van escribiéndose en el resto de discos. El problema de este nivel de RAID radica en la necesidad de calcular la paridad del dato, lo cual es necesario hacer cuando se escribe o modifica un dato del disco. Este cálculo introduce un retraso en la escritura que afecta al rendimiento de forma determinante, el uso de hardware adecuado para el cálculo de la paridad es determinante en la reducción de este tiempo.

Este RAID se consideró como parte de la solución debido a su redundancia y aparente buen rendimiento con el aumento del número de discos.

2.2.1.4. RAID 6

Este nivel de RAID es similar al nivel RAID 5, con el añadido de que calcula un segundo bloque de paridad del dato. Esto le permite recuperarse del fallo de hasta dos discos. Ambos bloques de paridad son repartidos por todos los discos miembros. Al igual que el RAID 5, escrituras y lecturas se realizan en paralelo por lo que una cantidad mayor de discos en el *array*, por norma general, aumentará el rendimiento. El ratio de espacio total disponible en este nivel es $1 - (2/n)$, siendo n el número total de discos que forman el *array*.

A continuación, un diagrama explicativo de RAID 6:

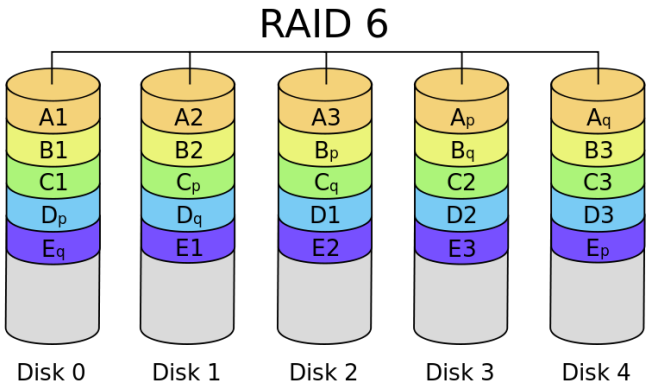

Ilustración 23: RAID 6. [22]

Se observa cómo, al igual que en el nivel RAID 5, el dato es dividido en *stripes*, y ambas paridades son escritas en otros dos discos, nunca en el mismo. Este cálculo añade una carga extra de cálculo que requiere ser atendida lo antes posible para evitar retrasos en las escrituras o modificaciones de datos que podrían ser inadmisibles en determinados contextos.

Este RAID se consideró como parte de la solución debido a su alta capacidad de tolerancia a fallos y aparente buen rendimiento con alto número de discos, pero la controladora hardware no lo soporta.

2.2.1.5. RAID 10

Este nivel de RAID se caracteriza por estar formado por un un RAID 0 dónde los miembros que forman el *array* son a su vez unidades lógicas creadas con un RAID 1 de dos discos. Permite el fallo de varios discos, uno por RAID 1, aunque expone al disco restante a un posible fallo total si también fallase con la consecuente pérdida de datos. Este nivel también divide el dato en *stripes*, cada cual es enviado a uno de los RAID 1, por lo que tanto en escrituras como en lecturas hay paralelización. El espacio disponible es la mitad del espacio total resultante de la suma del espacio de cada disco.

A continuación, un diagrama explicativo de RAID 10:

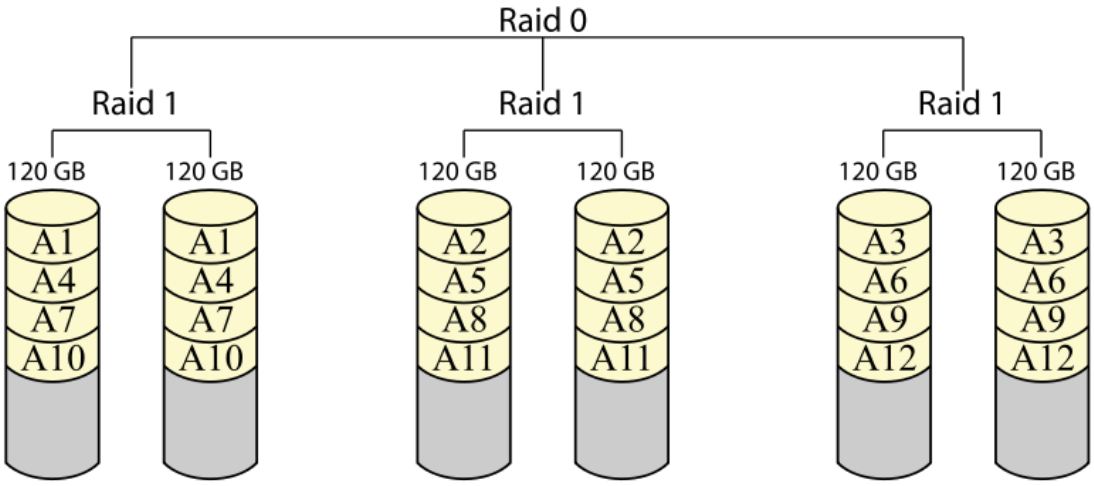

Ilustración 24: RAID 10. [23]

El dato es dividido en *stripes*, cada *stripe* es enviado a uno de los RAID 1, y escrito en ambos discos. Como se puede observar no se calcula paridad sino que la capacidad de soportar pérdidas de discos es a base de duplicar los *stripes* de cada dato.

Este RAID se consideró como parte de la solución debido a su redundancia y a la paralelización que aporta, la cual, sin duda, acelera el proceso de las operaciones.

2.2.2. Linux Buffer Cache / Page Cache

Esta sección explica, en general, el funcionamiento de Linux *page cache*, para más detalle consultar [24].

Actualmente, a partir de la versión del *kernel* de Linux 2.6, tanto *buffer cache* como *page cache* hacen referencia a la misma cache, aunque se utilizará el término *page cache* durante este documento por ser un nombre más cercano a la realidad actual del *kernel*. Previamente, en la versión 2.2 existían por separado.

Como su nombre indica, es una cache de páginas de memoria. Dichas páginas se crean a partir de operaciones de lectura y escrituras.

La *page cache* contiene páginas de datos accedidos recientemente por el sistema. Durante una operación de entrada y salida, el *kernel* Linux comprueba si el bloque físico de disco a acceder reside en alguna página de dicha cache. Si dicha comprobación devuelve un resultado positivo, entonces, el dato es accedido desde la cache rápidamente en vez de buscarlo en el disco, que es mucho más lento como se vio en la sección 2.1.1.1.

Concretamente, en operaciones de lectura mediante la llamada al sistema *read()*, si el dato se encuentra en *page cache* entonces se devuelve directamente sin necesidad de acceder a disco. En cambio, si el dato no está en la cache se accede a disco para leerlo provocando una espera mucho mayor y una pérdida grande de rendimiento, teniendo en cuenta los tiempos que se explicaron en la sección 2.1.1.

En el caso de operaciones de escritura mediante la llamada al sistema *write()*, las escrituras se delegan en la cache, la cual almacena el dato a escribir y marca la página como *dirty*. A partir de este momento, dicha página queda marcada para volcar su contenido en disco cuando sea necesario, *write-back*.

Es necesario volcar a disco las páginas marcadas como *dirty* en las siguientes situaciones:

- La cantidad de memoria libre en el sistema baja de cierto límite. El objetivo es conseguir más memoria libre para otros procesos.

- La cantidad de páginas marcadas como *dirty* sobrepasa cierto límite de tiempo, de este modo el *kernel* se asegura que no existen páginas en este estado indefinidamente.

2.2.2.1. Pdflush daemon

Este proceso está formado por un *pool* de hilos, el cual se encargan de volcar los datos a disco en el caso de que se dé como verdadera alguna de las dos condiciones

explicadas anteriormente. Disponer de varios hilos permite aprovechar la existencia de varios discos para paralelizar el volcado de datos.

Para la primera condición se utiliza un hilo de *pdflush* para comenzar a escribir un número mínimo predeterminado de páginas en disco. Dicho hilo escribe datos en disco hasta que se cumplen las siguientes dos condiciones:

- El número mínimo de páginas a escribir es alcanzado.
- La cantidad de memoria disponible supera de nuevo el límite establecido.

Para la segunda condición, *pdflush* despierta periódicamente para escribir aquellas páginas expiradas. Es importante, ya que si se produce un fallo en el sistema, aquellas páginas marcadas como *dirty* se pierden. Por lo tanto, no es seguro no consolidar las escrituras en disco durante demasiado tiempo.

En sistemas Linux es posible configurar los límites de las condiciones mencionadas así como el comportamiento de *pdflush*.

2.2.3. Sistema de ficheros

El sistema de ficheros es una abstracción, que ofrece una vista lógica de los datos al usuario y que le permite realizar una gestión de los mismos mediante la realización de operaciones bien definidas. Existen diferentes tipos de datos, como ficheros, directorios, enlaces simbólicos, enlace duros, tuberías y ficheros especiales que hacen referencia a dispositivos. [25]

Por lo tanto, gracias al sistema de ficheros, otros programas o usuarios podrán visualizar el contenido del sistema de almacenamiento secundario mediante una estructura lógica similar a la siguiente, dónde los datos se organizan en forma de árbol de directorios.

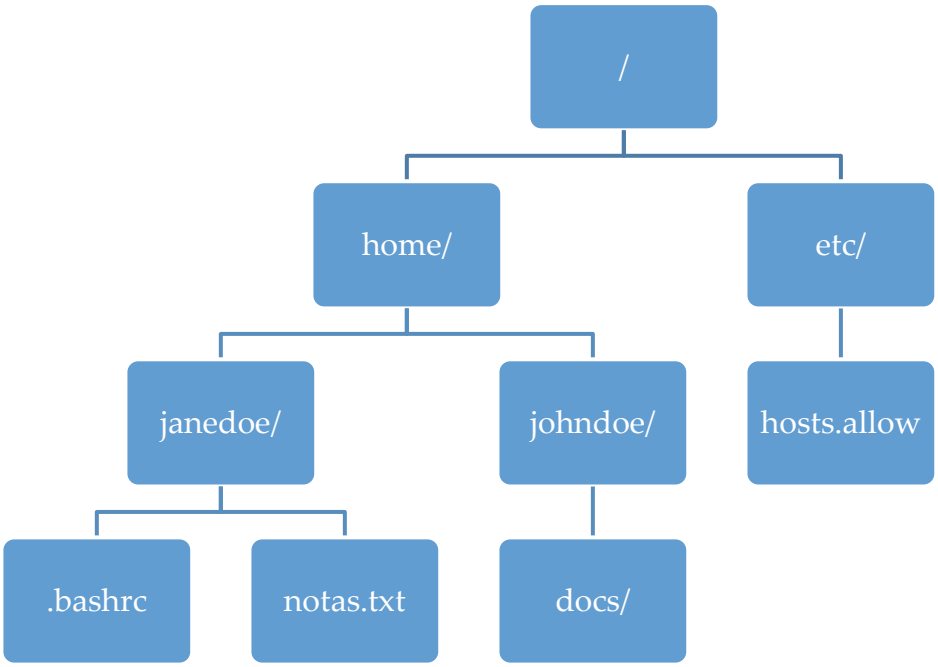

Ilustración 25: árbol de directorios.

En el diagrama anterior se observa una estructura lógica de ficheros y directorios, conforma una forma unívoca de referenciar cada elemento del sistema de ficheros mediante la siguiente notación.

/<directorio>/<directorio>/<fichero>

/etc/hosts.allow

/home/johndoe/docs/

Para lograr esto, los sistemas de ficheros incluyen estructuras de datos en los dispositivos de disco en forma de índices y tablas, los cuales relacionan los datos localizados físicamente en diferentes zonas del disco con su vista lógica en el árbol de directorios.

Existen multitud de tipos de sistemas de ficheros, todos ellos con diferentes características, sobre todo a nivel de funcionamiento interno. Están muy ligados al sistema operativo que se utiliza, el cual se encarga de abstraer a los programas de los sistemas de ficheros ofreciendo una interfaz común para todos ellos. [26]

En esta sección, se describen *grosso modo* diferentes sistemas de ficheros.

2.2.3.1. FAT32

El sistema de ficheros FAT32 fue desarrollado por Microsoft como mejora al sistema de ficheros FAT16 en el año 1996. Es un sistema de ficheros antiguo, que en su tiempo gozó de gran popularidad, nótese que las características, a continuación mencionadas, han quedado en la actualidad completamente desfasadas en comparación con sistemas de ficheros actuales.

FAT32 tiene las siguientes características principales [27]:

- Permite discos de hasta 2TB.
- Crea copias de seguridad de las estructuras de datos propias del sistema de ficheros, proporcionando cierta robustez ante posibles pérdidas de las estructuras originales.
- Gran rendimiento en comparación con su predecesor FAT16.

Quedó desfasado por sus limitaciones en los siguientes aspectos:

- Rápida degradación, requiriendo continuos mantenimientos para optimizar la localización de los datos en el disco, evitar pérdidas de rendimiento y envejecimiento prematuro del hardware.
- Incapaz de manejar grandes discos y ficheros.
- Estructuras de datos fácilmente corruptibles, causando problemas de fiabilidad.

En el sistema operativo GNU/Linux se desarrolló un módulo específico para soportar este sistema de ficheros, aunque nunca llegó a ser muy utilizado.

2.2.3.2. Ext2

El sistema de ficheros ext2 fue desarrollado por Rémy Card como mejora al sistema de ficheros ext en el año 1993. Es un sistema de ficheros antiguo, desarrollado

específicamente para el sistema operativo GNU/Linux, actualmente es un sistema desfasado aunque incluyó características interesantes, superiores algunas a FAT32, que han marcado la evolución de los sistemas de ficheros para GNU/Linux.

Ext2 tiene las siguientes características principales [25]:

- Permite discos de hasta 4TB.
- Nombres de fichero de hasta 255 bytes.
- Múltiples tamaños de bloque.
- Reserva de espacio para usuario *root*.
- Múltiples atributos para cada fichero o directorio, afecta al acceso a los mismos.
- Ext2 es pequeño en cuanto a líneas de código, fácil de mantener, robusto y fácil de comprender. Características muy importantes en el contexto del software libre dónde se desarrolló este sistema de ficheros.

Quedó desfasado por sus limitaciones en los siguientes aspectos:

- Requiere mucho tiempo ejecutar las comprobaciones necesarias cuando el sistema ha fallado. Estos tiempos se incrementan cuando la cantidad de estructuras de datos utilizadas por el sistema de ficheros es grande.
- No incluye un sistema transaccional. El sistema transaccional permite dotar al sistema de ficheros de la capacidad de identificar transacciones. De este modo, en caso de fallo del sistema, las operaciones no quedan incompletas en un estado inconsistente.

Ext2 fue reemplazado por ext3, el cual solucionó ambos problemas. La gestión del sistema transaccional, también llamado *journaling*, se realiza en una capa separada de ext3 aunque el sistema de ficheros trabaja con transacciones siendo capaz de identificarlas.

2.2.3.3. Ext4

El sistema de ficheros ext4 fue desarrollado en parte por la compañía Cluster File Systems, además de por algunos desarrolladores del *kernel* de Linux asociados al ya existente ext3. Ext4 fue liberado como estable en el año 2008 para sistemas operativos GNU/Linux, aunque otros sistemas como BSD también lo incluyen. Actualmente, es la versión más moderna del conjunto de sistema de ficheros ext.

Ext4 tiene las siguientes características principales:

- Permite discos de hasta 1EB, realmente no existen discos físicos de este tamaño en la actualidad, pero sí discos lógicos creados por *arrays* o conjuntos de servidores de almacenamiento que pueden alcanzar estos tamaños.
- *Checksums* en el sistema de *journaling* introducido en ext3. El cual se puede desactivar.
- Mejoras en la velocidad de las comprobaciones en caso de fallo del sistema con respecto a ext3.
- Es compatible hacia atrás con sistemas ext2 y ext3.

Actualmente, este sistema de ficheros es ampliamente utilizado en todo tipo de sistemas GNU/Linux. En cambio, el principal desarrollador de ext3 y ext4, Theodore Ts'o, opina que ext4 se basa en tecnología y conceptos antiguos y que, en realidad, la dirección correcta en el desarrollo de sistemas de ficheros en el futuro la llevan proyectos como BTRFS. Dicho sistema de ficheros está diseñado entorno a la idea de escalabilidad, fiabilidad y fácil gestión. En estos momentos no es considerado un desarrollo en fase estable para su uso en producción.

2.2.3.4. XFS

El sistema de ficheros XFS fue desarrollado por la compañía Silicon Graphics para sus propios sistemas en el año 1994 y posteriormente portado a sistemas GNU/Linux en el año 2000. Actualmente, es soportado por otros sistemas operativos como tipo BSD.

XFS tiene las siguientes características principales [28] [29]:

- Soporta discos de hasta 8EB. Como se comentó en el apartado anterior, no existen discos de este tamaño aunque sí dispositivos lógicos creados a partir de *arrays* o conjuntos de servidores de almacenamiento sobre los que operar.

- Especialmente diseñado para alto rendimiento en entornos de E/S paralela donde se dispone de varios dispositivos de almacenamiento.

- Sistema de *journaling*.

- Redimensionamiento en caliente del sistema de ficheros.

En cambio, tiene algunos inconvenientes a tener en cuenta en entornos especialmente cambiantes:

- No permite reducir el sistema de ficheros. Incluso existiendo espacio no utilizado en el mismo.

- Las operaciones sobre estructuras de datos que albergan los metadatos del sistema de ficheros son lentas. Estos problemas fueron mitigados por Red Hat cuyos cambios se aplicaron como estables al *kernel* de sistemas GNU/Linux en el año 2011, logrando asemejar los tiempos de operaciones sobre metadatos a los de ext4. [30]

Actualmente, a pesar de los buenos resultados de XFS, no llega a los niveles de implantación de ext4. En cambio, es especialmente utilizado en sistemas dónde las exigencias de almacenamiento son muy grandes, siendo solo superado por el ya comentado sistema de ficheros BTRFS.

2.2.4.NFS

NFS es un sistema de ficheros distribuido con sistema de recuperación de errores e independiente del protocolo de transporte y del sistema operativo utilizado, lo que propicia su utilización en entornos NAS heterogéneos.

Algunas de sus características generales en su versión 4 son [31]:

- XDR para representación externa de datos.

- RPC para llamada a procedimientos.

- Atributos, NFS garantiza que el servidor proporcionará un número mínimo de atributos de cada fichero a los clientes.

- Bloqueo de ficheros, permite evitar problemas en accesos concurrentes al mismo dato. Permite bloquear solo partes determinadas del fichero. De este modo, diversos clientes pueden acceder al mismo conjunto de datos con seguridad.

- Cache en el cliente, atributos e información de directorios es cacheada por el cliente. El contenido de ficheros también es cacheado, la validez de la información contenida en dicha cache es comprobada cuando se abre el fichero.

NFS es, en realidad, un protocolo de comunicación, sigue el modelo cliente-servidor. El servidor proporciona a la red a la que está conectado un interfaz con una serie de operaciones. Mediante estas operaciones, los clientes pueden realizar acciones sobre los ficheros del servidor. Gracias a este mecanismo, los clientes puede operar sobre un sistema de ficheros situado en el servidor como si estuviera albergado localmente y, además, sin tener en cuenta el sistema de ficheros real dónde el servidor almacena los datos.

En el contexto de NFS, se denomina *export*, a ofrecer una vista de una parte del sistema de ficheros del servidor a los clientes. Se pueden definir varios *export*, cada uno de ellos con diferentes atributos que definen la forma en la que los clientes podrán acceder a los datos.

A continuación, un diagrama que representa un *export* de NFS:

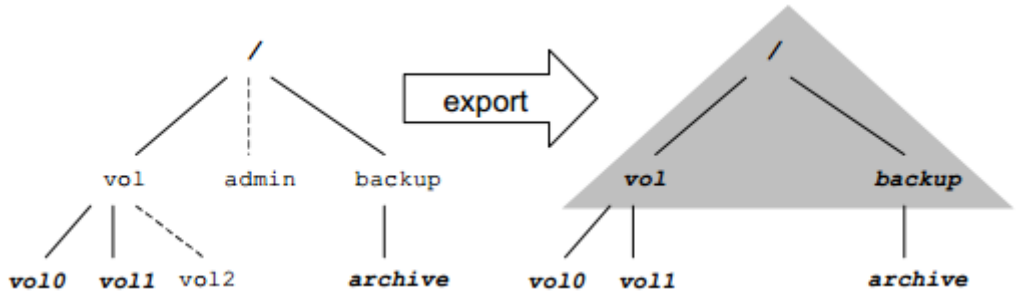

Ilustración 26: NFS export. [32]

Se muestra como el sistema de ficheros del servidor (a la izquierda) es exportado a un cliente omitiendo los ficheros *admin* y *vol2*.

Además, NFS permite configurar la forma en la cual se asignan UIDs y GIDs a los ficheros en el servidor, ya que la gestión de los mismos es independiente entre un cliente y otro. Este tipo de problemas también se soluciona con un sistema de autenticación común para los clientes, de modo que un mismo usuario tiene el mismo UID independientemente del cliente desde el que se conecte.

Cabe mencionar que NFS trabaja con la interfaz que provee *Linux Virtual File System*.

2.2.5. Linux Virtual File System

Es una interfaz ofrecida por el *kernel* de Linux que permite realizar operaciones sobre los sistemas de ficheros, de forma completamente transparente e independientemente del sistema de ficheros real utilizado en cada dispositivo de bloques.

Además, ofrece la abstracción de fichero y directorio. Gracias a ambas se dispone del conocido árbol de directorios, dónde es posible montar más dispositivos de bloques.

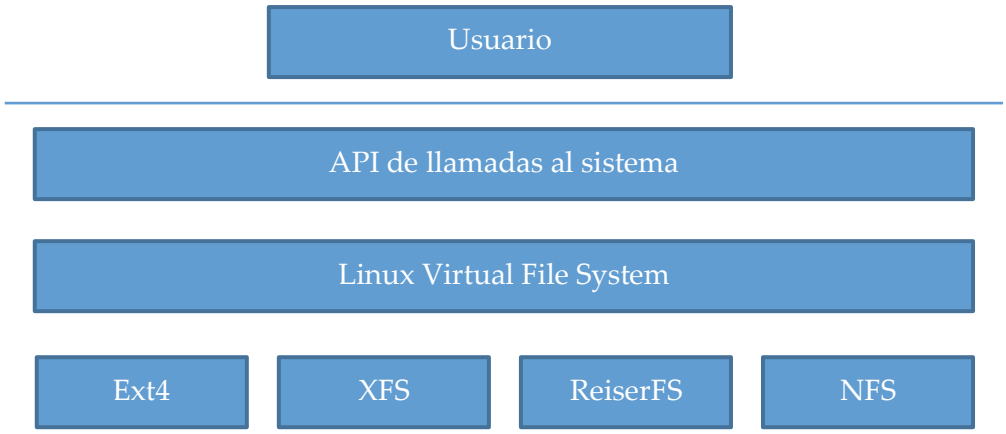

Ilustración 27: Linux Virtal File System.

En la ilustración anterior, la línea delimita el espacio de usuario donde se encuentran las aplicaciones y el espacio de *kernel*. Por lo tanto, las aplicaciones pueden ordenar la lectura o escritura de datos utilizando el API de llamadas al sistema, dichas llamadas

utilizan el interfaz *Linux Virtual File System* para ejecutar las operaciones y no saben qué sistema de ficheros se utiliza.

2.3. Herramientas de benchmarking

Estas herramientas ejecutan diversos tipos de pruebas de forma automática sobre sistemas de ficheros. Habitualmente, son capaces de medir diversos aspectos que posteriormente presentan a modo de informe. Intentan ejecutar pruebas que representan los tipos de operaciones que pueden darse en un sistema.

Los tipos de herramientas de benchmarking que se han contemplado en este proyecto son de tipo sintético, no simulan una carga real pero sí dan una idea de la capacidad máxima del sistema.

2.3.1. Bonnie++

Esta herramienta realiza pruebas de ancho de banda durante operaciones de entrada y salida básicas.

Las principales características de Bonnie++ son:

- Realiza pruebas dónde ficheros de distintos tamaño son leídos y escritos.
- Realiza pruebas de creado, borrado y petición de metadatos de ficheros.
- Las pruebas las realiza en orden secuencial y aleatorio.

Esta herramienta podría servir para las pruebas, pero no tiene todas las funcionalidades deseables para un *benchmark* en profundidad en el que se pueda generar gran cantidad de información.

2.3.2. IOzone

Esta herramienta realiza pruebas basadas en un amplio número de tipos de operaciones entrada y salida, es una herramienta compleja con numerosos parámetros de configuración y capaz de automatizar más acciones durante las pruebas como el vaciado de *buffers* entre prueba y prueba.

Las principales características de IOzone son:

- Trece tipos de operaciones.
- Mide ancho de banda durante operaciones.
- Mide latencia durante operaciones.
- Múltiples tamaños de fichero y de transferencia de datos durante las operaciones.
- La información se puede exportar a Excel.
- Bien documentado.

Esta herramienta es una buena candidata, destaca sobre todo por la gran variedad de pruebas que realiza y la forma en la que los datos son ofrecidos facilitando el estudio de los mismos.

Capítulo 3

3. Análisis

Este capítulo recoge la segunda fase del proyecto, se detallan los requisitos de usuario identificados. Los requisitos se dividen en dos grupos, requisitos de capacidad y requisitos de restricción.

Además, la definición de todos los requisitos cumplirá con el formato de la siguiente tabla.

ID			
Nombre			
Descripción			
Necesidad		**Prioridad**	
Estabilidad		**Claridad**	

Tabla 1: ejemplo de requisito.

El significado de los campos de dicha tabla es el siguiente:

- **ID**: identificador del requisito, sigue el siguiente patrón REQ-C-XX para los requisitos de capacidad y REQ-R-XX en el caso de los requisitos de restricción.
- **Nombre**: nombre representativo del requisito.
- **Descripción**: descripción breve del requisito.
- **Necesidad**: indica la necesidad que tiene el sistema de dicho requisito. Sus valores son:
  - Esencial
  - Deseable
  - Opcional

- **Prioridad**: importancia de requisito, sus valores son:
  - Alta
  - Media
  - Baja

- **Estabilidad**: indica cómo de propenso es el requisito a sufrir cambios durante el desarrollo del proyecto, sus valores son:
  - Estable
  - No estable

- **Claridad**: establece si el requisito está claramente definido o no, sus valores son:
  - Alta
  - Media
  - Baja

3.1. Requisitos de capacidad

Los requisitos de capacidad definen qué capacidades o funcionalidades debe cumplir el sistema.

ID	**REQ-C-01**		
Nombre	Escritura en almacenamiento secundario.		
Descripción	Los clientes GNU/Linux de la red podrán escribir en el almacenamiento secundario proporcionado por el sistema.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Alta	**Claridad**	Alta

Tabla 2: requisito de usuario REQ-C-01

ID	**REQ-C-02**		
Nombre	Lectura en almacenamiento secundario.		
Descripción	Los clientes GNU/Linux de la red podrán leer en el almacenamiento secundario proporcionado por el sistema.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Alta	**Claridad**	Alta

Tabla 3: requisito de usuario REQ-C-02

ID	**REQ-C-03**		
Nombre	Software libre.		
Descripción	El sistema será creado con productos con licencias de software libre.		
Necesidad	Deseable	**Prioridad**	Alta
Estabilidad	Alta	**Claridad**	Media

Tabla 4: requisito de usuario REQ-C-03

ID	**REQ-C-04**		
Nombre	Delegación de gestión del sistema de ficheros al servidor.		
Descripción	Los clientes del sistema podrán delegar por completo el mantenimiento del sistema de ficheros en el sistema.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Alta	**Claridad**	Alta

Tabla 5: requisito de usuario REQ-C-04

ID	**REQ-C-05**		
Nombre	Integración sencilla.		
Descripción	Los clientes podrán comenzar a utilizar el almacenamiento secundario aplicando el menor número de cambios posibles.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Alta	**Claridad**	Media

Tabla 6: requisito de usuario REQ-C-05

ID	**REQ-C-06**		
Nombre	Tolerancia a fallos.		
Descripción	Los clientes podrán seguir utilizando el sistema aunque se produzcan fallos de hardware comunes, como el fallo de discos o de fuentes de alimentación.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Alta	**Claridad**	Alta

Tabla 7: requisito de usuario REQ-C-06

ID	**REQ-C-07**		
Nombre	Usuarios.		
Descripción	El sistema de almacenamiento debe ser capaz de almacenar la información referente a la propiedad de los ficheros y directorios creados por los clientes GNU/Linux.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Alta	**Claridad**	Alta

Tabla 8: requisito de usuario REQ-C-07

ID	REQ-C-08		
Nombre	Grupos de usuarios.		
Descripción	El sistema de almacenamiento debe ser capaz de almacenar la información referente al grupo de usuarios de los ficheros y directorios creados por los clientes GNU/Linux.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Alta	**Claridad**	Alta

Tabla 9: requisito de usuario REQ-C-08

ID	REQ-C-09		
Nombre	Permisos		
Descripción	El sistema de almacenamiento debe ser capaz de almacenar la información referente a los permisos asignados a los ficheros y directorios creados por los clientes GNU/Linux.		
Necesidad	Esencial	**Prioridad**	Alta
Estabilidad	Alta	**Claridad**	Alta

Tabla 10: requisito de usuario REQ-C-09

ID	REQ-C-10		
Nombre	Accesibilidad		
Descripción	El sistema debe permitir el acceso a los datos de los clientes GNU/Linux por parte de terceros en caso de ser necesario, como por ejemplo, un servidor de backups o profesores sin necesidad de que los clientes GNU/Linux estén en funcionamiento.		
Necesidad	Deseable	**Prioridad**	Alta
Estabilidad	Alta	**Claridad**	Alta

Tabla 11: requisito de usuario REQ-C-10

ID	REQ-C-11		
Nombre	Flexibilidad de espacio de disco.		
Descripción	El sistema debe garantizar la posibilidad de aumentar el espacio disponible para datos en un futuro.		
Necesidad	Esencial	Prioridad	Alta
Estabilidad	Alta	Claridad	Alta

Tabla 12: requisito de usuario REQ-C-11

ID	REQ-C-12		
Nombre	Integridad de datos.		
Descripción	El sistema debe ser capaz de realizar comprobaciones de integridad de datos de forma automática.		
Necesidad	Esencial	Prioridad	Media
Estabilidad	Alta	Claridad	Alta

Tabla 13: requisito de usuario REQ-C-12

3.2. Requisitos de restricción

Los requisitos de restricción son aquellos que definen las limitaciones al funcionamiento del sistema o a su diseño.

ID	REQ-R-01		
Nombre	Seguridad de acceso a datos.		
Descripción	El sistema no permite el acceso a datos cuando la configuración de usuario, grupo y permisos no lo permite.		
Necesidad	Esencial	Prioridad	Alta
Estabilidad	Alta	Claridad	Alta

Tabla 14: requisito de restricción REQ-R-01

ID	**REQ-R-02**		
Nombre	Alta disponibilidad		
Descripción	El sistema debe estar en funcionamiento bajo un régimen 24x7, salvo fallos no comunes de hardware.		
Necesidad	Deseable	**Prioridad**	Media
Estabilidad	Alta	**Claridad**	Alta

Tabla 15: requisito de restricción REQ-R-02

ID	**REQ-R-03**		
Nombre	Acceso seguro		
Descripción	El acceso remoto al sistema debe ser seguro y no permitir el acceso a equipos no pertenecientes a un determinado grupo de confianza.		
Necesidad	Esencial	**Prioridad**	Baja
Estabilidad	Alta	**Claridad**	Alta

Tabla 16: requisito de restricción REQ-R-03

3.3. Casos de uso

En esta sección se detallan los casos de uso del sistema. Cada caso de uso se compone de una especificación en forma de tabla y un diagrama.

Además, la especificación cumplirá con el formato de la siguiente tabla.

ID	
Nombre	
Descripción	
Pre-condiciones	
Post-condiciones	

Tabla 17: ejemplo de especificación de caso de uso.

El significado de los campos de dicha tabla es el siguiente:

- **ID**: identificador del caso de uso, sigue el siguiente patrón CU-XX.
- **Nombre**: nombre representativo del caso de uso.
- **Descripción**: descripción breve del caso de uso.
- **Pre-condiciones**: condiciones necesarias que se deben dar para la realización del caso de uso.
- **Post-condiciones**: condiciones que se dan al realizarse el caso de uso.

A continuación, los casos de uso del sistema.

ID	CU-01
Nombre	Montar almacenamiento secundario.
Descripción	▪ El sistema cliente solicita al servidor de almacenamiento montar un sistema de ficheros. ▪ El sistema de almacenamiento comprueba que el cliente tenga permiso para montar dicho sistema de ficheros. ▪ El sistema de almacenamiento contesta al cliente el resultado de la operación. ▪ El cliente monta el sistema de ficheros remoto en un directorio de su sistema local.
Pre-condiciones	▪ El servidor de almacenamiento debe estar iniciado. ▪ El servidor de almacenamiento debe estar correctamente configurado. ▪ El cliente debe estar conectado a la red del servidor.
Post-condiciones	El sistema cliente monta en su árbol del sistema de ficheros local el sistema de ficheros remoto, permitiendo el acceso a su contenido en función de los permisos configurados en el mismo.

Tabla 18: caso de uso CU-01.

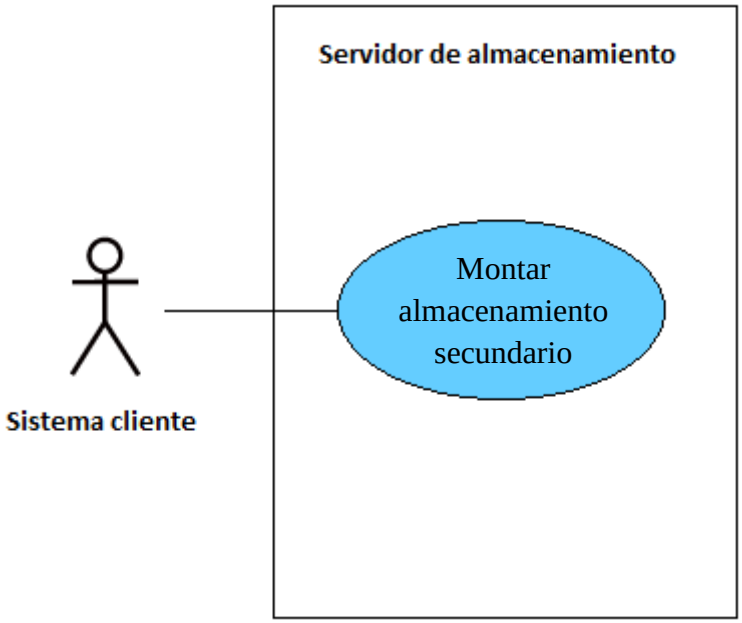

Ilustración 28: caso de uso CU-01.

ID	CU-02
Nombre	Desmontar almacenamiento secundario.
Descripción	▪ El sistema cliente solicita al servidor de almacenamiento desmontar un sistema de ficheros. ▪ El sistema de almacenamiento contesta al cliente el resultado de la operación. ▪ El cliente desmonta el sistema de ficheros remoto de su árbol de directorios local.
Pre-condiciones	▪ El servidor de almacenamiento debe estar iniciado. ▪ El servidor de almacenamiento debe estar correctamente configurado. ▪ El cliente debe estar conectado a la red del servidor.
Post-condiciones	El sistema de ficheros remoto es desmontado del árbol del sistema de ficheros del cliente.

Tabla 19: caso de uso CU-02.

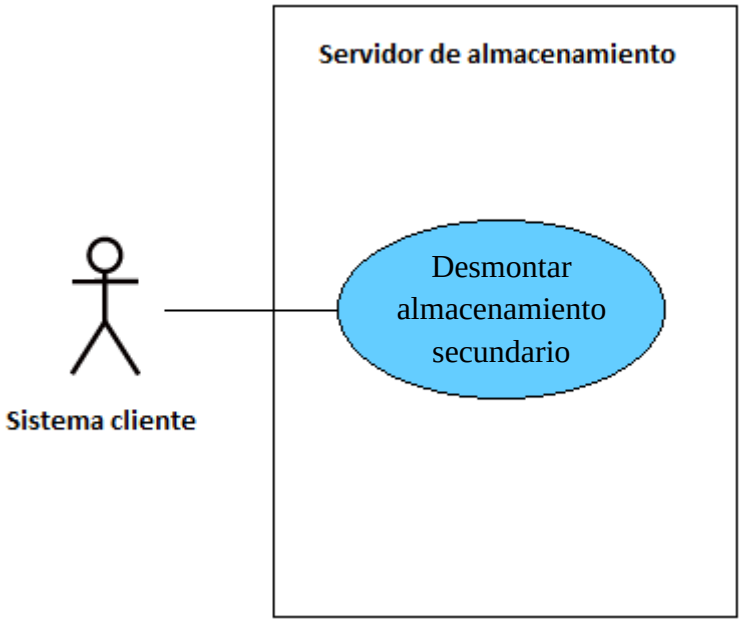

Ilustración 29: caso de uso CU-02.

ID	**CU-03**
Nombre	Abrir fichero.
Descripción	▪ El sistema cliente solicita al servidor de almacenamiento abrir el fichero. ▪ El sistema de almacenamiento comprueba que el cliente tenga permiso para acceder al fichero, anota qué cliente lo solicita y realiza la llamada al sistema *open*. ▪ El sistema de almacenamiento contesta al cliente el resultado de la operación. ▪ El cliente abre el fichero.
Pre-condiciones	▪ El servidor de almacenamiento debe estar iniciado. ▪ El servidor de almacenamiento debe estar correctamente configurado. ▪ El cliente debe estar conectado a la red del servidor. ▪ El sistema de ficheros remoto, dónde está localizado el fichero, debe estar montado en el cliente que realiza la operación.
Post-condiciones	El fichero queda marcado como abierto.

Tabla 20: caso de uso CU-03.

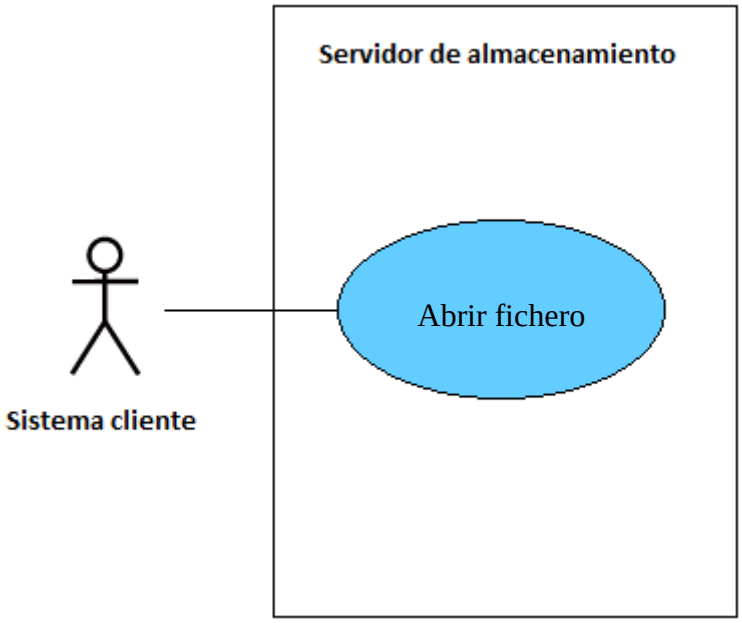

Ilustración 30: caso de uso CU-03.

ID	**CU-04**
Nombre	Cerrar fichero.
Descripción	▪ El sistema cliente solicita al servidor de almacenamiento cerrar el fichero. ▪ El sistema de almacenamiento comprueba que el cliente tenga permiso para acceder al fichero y realiza la llamada al sistema *close*. ▪ El sistema de almacenamiento contesta al cliente el resultado de la operación. ▪ El cliente cierra el fichero.
Pre-condiciones	▪ El servidor de almacenamiento debe estar iniciado. ▪ El servidor de almacenamiento debe estar correctamente configurado. ▪ El cliente debe estar conectado a la red del servidor. ▪ El sistema de ficheros remoto dónde está localizado el fichero debe estar montado en el cliente que realiza la operación.
Post-condiciones	El fichero queda marcado como cerrado.

Tabla 21: caso de uso CU-04.

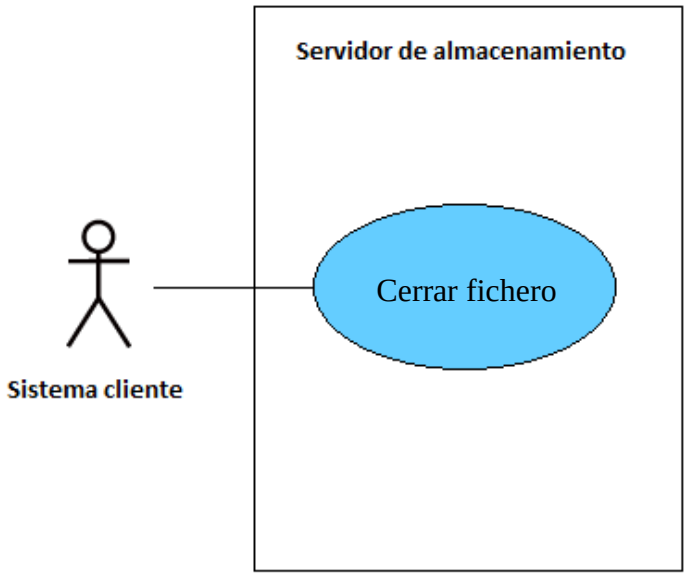

Ilustración 31: caso de uso CU-04.

ID	**CU-05**
Nombre	Leer dato.
Descripción	▪ El sistema cliente solicita al servidor de almacenamiento cuantos bytes quiere leer, la posición de dichos bytes y la localización del fichero dónde se encuentran. ▪ El sistema de almacenamiento comprueba que el cliente tenga permiso para acceder al fichero, que el fichero esté abierto previamente y realiza las llamadas al sistema *seek* y *read* para poder leer la información. ▪ El sistema de almacenamiento contesta al cliente el resultado de la operación. ▪ El cliente almacena el dato leído.
Pre-condiciones	▪ El servidor de almacenamiento debe estar iniciado. ▪ El servidor de almacenamiento debe estar correctamente configurado. ▪ El cliente debe estar conectado a la red del servidor. ▪ El sistema de ficheros remoto dónde está localizado el fichero debe estar montado en el cliente que realiza la operación.
Post-condiciones	El sistema cliente obtiene una copia del dato leído del fichero.

Tabla 22: caso de uso CU-05.

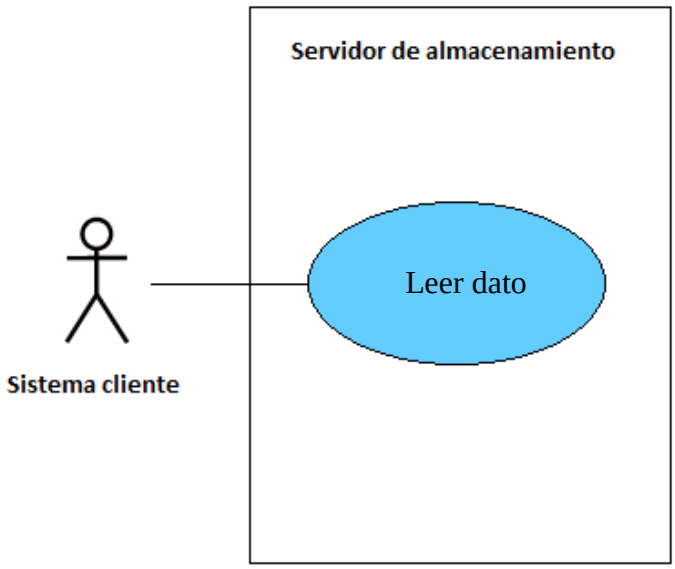

Ilustración 32: caso de uso CU-05.

ID	**CU-06**
Nombre	Escribir dato.
Descripción	▪ El sistema cliente solicita al servidor de almacenamiento cuantos bytes quiere escribir, la posición dónde se desea escribir, el dato y la localización del fichero dónde se desea escribir. ▪ El sistema de almacenamiento comprueba que el cliente tenga permiso para acceder al fichero, que el fichero esté abierto previamente y realiza las llamadas al sistema *seek* y *write* para poder escribir el dato en la posición indicada. ▪ El sistema de almacenamiento contesta al cliente el resultado de la operación. ▪ El cliente da por escrito el dato.
Pre-condiciones	▪ El servidor de almacenamiento debe estar iniciado. ▪ El servidor de almacenamiento debe estar correctamente configurado. ▪ El cliente debe estar conectado a la red del servidor. ▪ El sistema de ficheros remoto dónde está localizado el fichero debe estar montado en el cliente que realiza la operación.
Post-condiciones	El dato se escribe en el fichero destino y el sistema cliente da por escrito el dato.

Tabla 23: caso de uso CU-06.

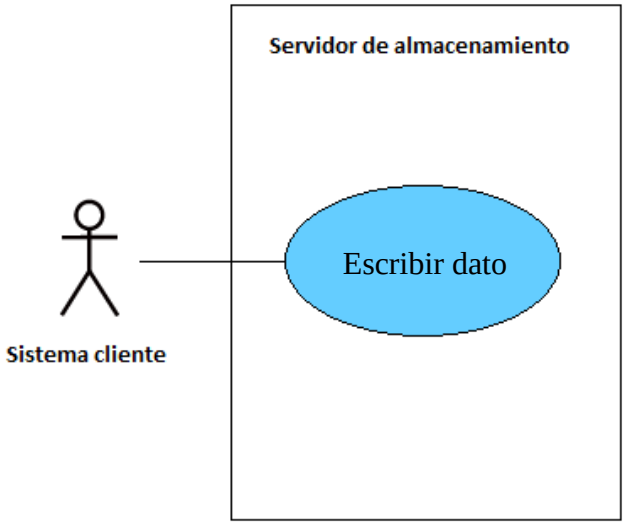

Ilustración 33: caso de uso CU-06.

Capítulo 4

4. Estudio del sistema actual y tecnologías relacionadas

Este capítulo recoge la tercera fase del proyecto, se detallan los objetivos a conseguir, la metodología general seguida para la realización del estudio, el diseño específico de cada estudio, tanto para el estudio del sistema actual como para el estudio de las tecnologías relacionadas, los resultados y finalmente las conclusiones.

4.1. Objetivos

El sistema a realizar es complejo, intervienen diferentes productos de diferentes fabricantes en las diferentes capas y componentes que forman el sistema. A pesar de que existen estudios teóricos e información sobre el rendimiento de cada una de las tecnologías por separado, no se dispone de información sobre el rendimiento de varios componentes funcionando de forma conjunta.

La realización de ambos estudios es necesaria porque, a pesar de que los sistemas operativos están optimizados para funcionar mejor con las operaciones más comunes, sabiendo el uso que se le dará al sistema, podemos adaptar algunos aspectos del diseño para que su rendimiento sea, incluso, mejor.

Además, antes de realizar un diseño que se adapte a los requisitos y proceder a crear el sistema, es necesario conocer cada producto propiamente dicho.

Por lo tanto, se ha pretendido:

- Adquirir conocimiento técnico sobre el conjunto de las tecnologías a utilizar.
- Restringir a un conjunto razonable la cantidad de posibles opciones que podrían teóricamente satisfacer los requisitos del proyecto.
- Obtener datos de forma empírica, con los productos reales, sobre rendimiento de diferentes posibles diseños.

- Alcanzar un conjunto de conclusiones que, junto con los requisitos y el estudio teórico ya realizado en la sección 2, apoyen una línea a seguir en el diseño del sistema.

4.2. Descripción general de la metodología

En esta sección se explica de forma resumida la metodología desarrollada para la realización del estudio del sistema actual y el estudio de rendimiento. La metodología pretende abarcar un conjunto razonable de posibles opciones eligiendo aquellas que, por sus características, representan mejor una posible solución a los requisitos.

Dicha metodología está compuesta por los siguientes pasos.

- Despliegue del hardware.
- Instalación del software necesario.
- Diseño y ejecución de los estudios.
- Obtención y análisis de los resultados.

En las siguientes secciones se detallan los estudios realizados.

4.3. Diseño del estudio

En esta sección se detallan las pruebas realizadas. Existen multitud de posibles configuraciones que podrían satisfacer en mayor o menor medida los requisitos, por lo que se han contemplado aquellas que mejor se ajustan a algunas de las necesidades.

Además, las configuraciones no contemplan posibles optimizaciones muy específicas sino, únicamente, configuraciones generales que pretenden dar una idea general de la mejor aproximación al diseño final del sistema.

Se han creado dos tipos de estudios:

- Comportamiento de un sistema similar, existe esta posibilidad ya que en la Universidad Rey Juan Carlos se dispone de acceso a un sistema que ofrece

almacenamiento secundario por red a las aulas GNU/Linux. Ha permitido obtener información y compararlo con el trabajo realizado en este proyecto.

- Comportamiento de las configuraciones potencialmente válidas de las tecnologías planteadas en la sección 2.

4.3.1. Comportamiento de aulas GNU/Linux

Las aulas GNU/Linux de la Universidad Rey Juan Carlos utilizan un sistema de almacenamiento secundario por red para acceder a los datos de los usuarios. Es interesante hacer un estudio de dicho sistema para conocer el comportamiento de un entorno de este tipo.

Este sistema dispone de un servidor de almacenamiento con las siguientes características:

Kernel	**Linux 3.0.0-22-server**
Distribución	Ubuntu 12.04 LTS
Dispositivos de bloque	▪ sda1 ▪ sdb1 ▪ sdc1 ▪ sdd1
Dispositivos raid	▪ md0 : active raid0 sdd1[2] sdc1[1] sdb1[0]
Sistema de ficheros local para raid	▪ ext3 ▪ Espacio disponible 5,4 TB.
Protocolo de sistema de ficheros por red	▪ NFS, ver sección 2.2.4.

Tabla 24: almacenamiento aulas GNU/Linux de la URJC.

Como se observa en la tabla anterior, el sistema dispone de un RAID de tipo 0 formado por tres dispositivos de bloque que son *sdb1*, *sdc1* y *sdd1*. Además, el servidor exporta el sistema de ficheros a los clientes mediante NFS.

No ha sido necesario despliegue de hardware ni de software para realizar este estudio porque el sistema existe en producción.

4.3.1.1. Tamaño de ficheros

Se ha tomado una muestra del tamaño de cada fichero en el directorio */home* del servidor de almacenamiento y se ha creado un diagrama *boxplot* utilizando el paquete estadístico *R*.

La captura se ha realizado utilizando el siguiente comando *bash*:

```
:/home# find . -type f -print0 | xargs -0 du -ahb > muestrahomes.txt
```

Posteriormente, se han eliminado de la muestra aquellos ficheros que ocupan 0 bytes y se ha utilizado *R* para generar la gráfica *boxplot* con los siguientes comandos:

```
> input <- scan("muestrahome.txt", what=list(bytes=0))

> summary(input$bytes)

> jpeg("boxplotchart.jpg")

> boxplot(input, log="y", main="Tamaño de ficheros en /home/",
xlab="1", ylab="Tamaño (bytes)")
```

4.3.1.2. Estadísticas de uso de disco

En esta sección se explica detalladamente el fichero */proc/diskstats* de los sistemas basados en GNU/Linux. Este fichero se ha utilizado en el estudio de carga del servidor actual durante su uso, ver sección 4.3.1.3.

Desde la versión del *kernel* de Linux 2.4.20 se han introducido estadísticas de disco más completas que ayuden a medir la actividad de las operaciones. En los *kernel* 2.6, esta información puede ser consultada en el fichero */proc/diskstats*.

Muestra información estadística relativa al uso de los discos y sus particiones.

La información contenida en el fichero aparece en forma de texto como se muestra a continuación:

```
8       0 sda 52458 57019 1167574 198530 104040 1550239 13246896 33706380 0 948290 33904950
8       1 sda1 51988 56421 1159042 197290 103653 1550239 13246896 33706380 0 947560 33903710
8       2 sda2 2 0 4 20 0 0 0 0 0 20 20
8       5 sda5 123 31 1232 590 0 0 0 0 0 590 590
8      16 sdb 1655 2910 18940 1030 0 0 0 0 0 1030 1030
8      17 sdb1 1448 2694 15556 570 0 0 0 0 0 570 570
```

Ilustración 34: /proc/diskstats.

Los datos aparecen en los campos a continuación del nombre de cada disco (*sda*, *sdb*, etc…) o partición (*sda1*, *sdb1*, etc…).

Los campos asociados tanto a discos como a particiones son los mismos y tienen el siguiente significado:

- **Campo 1**: número de lecturas completadas satisfactoriamente.
- **Campo 2**: número de lecturas combinadas. Dos lecturas adyacentes entre sí pueden ser realizadas en una sola operación. De modo que dos lecturas de 4 KB pueden convertirse en una sola de 8 KB.
- **Campo 3**: número de sectores leídos satisfactoriamente.
- **Campo 4**: número total de milisegundos empleados en lecturas.
- **Campo 5**: número de escrituras completadas satisfactoriamente.
- **Campo 6**: número de escrituras combinadas. Dos escrituras adyacentes entre sí pueden ser realizadas en una sola operación. De modo que dos escrituras de 4 KB pueden convertirse en una sola de 8 KB.
- **Campo 7**: número de sectores escritos satisfactoriamente.
- **Campo 8**: número total de milisegundos empleados en escrituras.
- **Campo 9**: número de operaciones de entrada y salida actualmente en progreso.

- **Campo 10**: número de milisegundos empleados en operaciones de entrada y salida. Este campo aumenta mientras el campo 9 no vale 0.

- **Campo 11**: número de milisegundos total empleado haciendo operaciones de entrada y salida. La diferencia con el campo 10 es que este solo se actualiza cuando comienza, se termina o se mezcla una operación de entrada y salida.

4.3.1.3. Carga del servidor durante su uso

Se ha recogido información para conocer el comportamiento del servidor durante su uso en horario de clases cuando los alumnos trabajan en sus asignaturas.

La información recogida es ofrecida por el sistema en el fichero */proc/diskstats*, como dicho fichero ofrece información para un instante determinado se debe volcar el contenido de dicho fichero cada cierto tiempo, en un fichero diferente cada vez.

Para ello se ha introducido en el *crontab* del sistema la siguiente línea:

```
* * * * * cat /proc/diskstats > ~/diskstats_`date +%s`.txt
```

De este modo se ha creado un fichero cada minuto con el contenido de */proc/diskstats*.

4.3.2. Comportamiento de las tecnologías estudiadas

En esta sección se muestra el comportamiento de diferentes tecnologías involucradas en la creación de un sistema de almacenamiento por red. El conjunto de las tecnologías estudiadas afectan a las siguientes capas del sistema:

- Hardware de almacenamiento.

- Sistema de ficheros.

- Sistema de ficheros remoto (NFS).

4.3.2.1. IOzone Filesystem Benchmark

En esta sección se muestra la herramienta IOzone como la elegida para realizar el *benchmark* de rendimiento. Es una herramienta de pruebas para sistemas de ficheros. Genera y mide diversas operaciones sobre ficheros. Es una herramienta ampliamente utilizada que funciona sobre diversas arquitecturas y sistemas operativos.

La herramienta es capaz de realizar las siguientes operaciones sobre ficheros [33]:

- **Write**: mide el rendimiento de la escritura de un fichero nuevo.
- **Re-write**: mide el rendimiento de la escritura en un fichero existente.
- **Read**: mide el rendimiento de la lectura de un fichero existente.
- **Re-read**: mide el rendimiento de la lectura de ficheros que fueron recientemente leídos.
- **Random read**: mide el rendimiento de la lectura realizada en una localización aleatoria de un fichero.
- **Random write**: mide el rendimiento de la escritura realizada en una localización aleatoria de un fichero.
- **Backward read**: mide el rendimiento de lecturas hechas hacia atrás en un fichero.
- **Record Re-write**: mide el rendimiento de escribir y re-escribir un dato en la misma posición de un fichero ya existente.
- **Stride read**: mide el rendimiento de patrones de lectura. Un ejemplo sería leer en un fichero 4 KB, avanzar 200 KB dentro del fichero, leer otros 4 KB y volver a avanzar 200 KB de forma sucesiva. Este tipo de lecturas puede producir cuellos de botella en un dispositivo particular si los datos fueron distribuidos en diferentes dispositivos de bloques.
- **F-write**: mide el rendimiento de la escritura de un fichero nuevo utilizando la función de biblioteca *fwrite()*, dicha función realiza escrituras utilizando un buffer dentro del espacio de memoria destinado al usuario. Puede mejorar el

rendimiento reduciendo el número real de llamadas al sistema de escritura, a costa de aumentar el tamaño de las transferencias en las llamadas al sistema.

- **F-rewrite**: mide el rendimiento de la escritura en fichero existente utilizando la función de biblioteca *fwrite()*, dicha función realiza escrituras utilizando un buffer dentro del espacio de memoria destinado al usuario. Puede mejorar el rendimiento reduciendo el número real de llamadas al sistema de escritura, a costa de aumentar el tamaño de las transferencias en las llamadas al sistema.

- **F-read**: mide el rendimiento de la lectura de un fichero utilizando la función de biblioteca *fread()*, dicha función realiza lecturas utilizando un buffer dentro del espacio de memoria destinado al usuario. Puede mejorar el rendimiento reduciendo el número real de llamadas al sistema de lectura, a costa de aumentar el tamaño de las transferencias en las llamadas al sistema.

- **F-reread**: mide el rendimiento de la re-lectura de un fichero que ha sido recientemente leído utilizando la función de biblioteca *fread()*, dicha función realiza lecturas utilizando un buffer dentro del espacio de memoria destinado al usuario. Puede mejorar el rendimiento reduciendo el número real de llamadas al sistema de lectura, a costa de aumentar el tamaño de las transferencias en las llamadas al sistema.

Además de realizar las operaciones definidas, IOzone utiliza diferentes tamaños de operaciones sobre ficheros de distinto tamaño. Esto es así debido a que el resultado de las operaciones varía en función de si las operaciones pueden realizarse íntegramente en la cache de la CPU, en la cache de memoria o si la operación excede por completo cualquier cache intermedia y realizan accesos reales a discos físicos. El tratamiento de las caches de lectura y escritura de GNU/Linux se ha explicado más detalladamente en la sección 2.2.2 de este documento.

IOzone devuelve una salida fácilmente leíble con otras herramientas. A continuación un ejemplo de salida de los resultados que arroja la herramienta.

						random	random	bkwd	record	stride				
KB	reclen	write	rewrite	read	reread	read	write	read	rewrite	read	fwrite	frewrite	fread	freread
64	4	673098	1648808	3057153	4274062	3203069	1638743	2772930	2133730	3363612	763021	1526887	3203069	4274062
64	8	763021	1828508	5389653	5283570	4564786	2006158	3588436	2379626	4207076	901377	1679761	3738358	4988978
64	16	821391	1828508	5283570	5860307	4897948	2067979	3541098	2358717	4018152	1049372	1879725	3738358	5389653
64	32	842003	1879725	4897948	5283570	4897948	2203800	3588436	2379626	3791156	1330167	2537060	3738358	5283570
64	64	877797	1947927	5389653	5389653	4897948	2203800	3541098	2298136	4207076	983980	1933893	4018152	4897948
128	4	747933	1705404	4267461	4407601	3536566	1778862	3199360	2286447	3199360	780556	1561553	3657016	4267461
128	8	841746	1939522	5122535	5325799	4596273	2132084	3867787	2727923	5122535	895074	1755594	4407601	5122535
128	16	889145	1967960	5122535	5122535	4934216	2211113	3867787	2621367	4557257	969421	1827299	4267461	5325799
128	32	914904	2035099	4934216	5074121	4934216	2326073	4135958	2608629	4267461	1196221	2132084	4407601	5122535
128	64	908709	2027414	5122535	5379161	5122535	2377579	4267461	2558895	4557257	1391557	2905056	4407601	5325799
128	128	940549	1885042	5122535	5325799	5122535	2409592	4267461	2464907	4759253	1067750	2058509	4407601	5122535
256	4	794889	1752173	4197489	4332998	3511189	1790149	3326279	2305128	3368013	778183	1532153	3935921	4281170
256	8	859127	1985448	5117791	5117791	4572895	2165650	4197489	2812272	4281170	879535	1729594	4652146	5117791
256	16	917880	2015259	4907284	5142301	4755158	2242541	4264168	2726577	4929815	952062	1778290	4496299	5022044
256	32	937924	2097948	4929815	5022044	4929815	2350543	4496299	2639446	4572895	1053922	1938842	4477549	5117791
256	64	965763	2114473	5022044	5217259	5117791	2435861	4572895	2719671	4840911	1237306	2285501	4652146	5117791
256	128	966633	2152625	5117791	5320671	5217259	2486631	4734192	2665657	4929815	1497953	3123106	4840911	5347168
256	256	955451	2187712	5217259	5347168	5320671	2533571	4840911	2557711	5022044	1132871	2187712	4840911	5320671
512	4	798675	1803387	4271001	4447925	3580298	1827947	3487274	2427068	3415178	792778	1570020	4061007	4262523
512	8	901241	1992458	5067142	5227492	4571003	2196227	4411377	2910635	4457157	885996	1736314	4742616	5127637

Ilustración 35: salida de datos de IOzone.

Los datos mostrados en la ilustración siguen el siguiente formato:

- **Primera columna**: tamaño del fichero sobre el que se ha realizado la prueba.
- **Segunda columna**: tamaño del dato utilizado (leído o escrito) durante la prueba.
- **Resto de columnas**: corresponden a cada tipo de operación descrita anteriormente. Los valores de estas columnas muestran el rendimiento en KB por segundo.

Nótese que la ilustración es solo un fragmento de una salida de IOzone real.

4.3.2.2. *Hardware y arrays de discos*

En la capa de hardware y *arrays* de discos se han seleccionado una serie de configuraciones presentadas en la siguiente tabla:

RAID	Número de discos	Stripe
5	3	64
5	3	128
5	3	256
5	3	512
5	3	1024
5	6	64
5	6	128
5	6	256
5	6	512
5	6	1024
5	12	64
5	12	128
5	12	256
5	12	512
5	12	1024
10	12	64
10	12	128
10	12	256
10	12	510
10	12	1024

Tabla 25: configuraciones de RAID a prueba.

- **RAID**: es el tipo de RAID a probar. Se han probado RAID 5 y RAID 10. Se han elegido ambos porque son aquellos, de los soportados por la controladora RAID utilizada, que ofrecen una relación entre espacio disponible, rendimiento y redundancia razonable teniendo en cuenta las características teóricas de ambos y los requisitos.

- **Número de discos**: es la cantidad de discos incluidos en el *array*. Es interesante variar este valor ya que potencialmente afecta al rendimiento del *array*, pudiendo convertir la opción del RAID 5 en interesante a pesar de su hándicap en escrituras como se explica en la sección 2.2.1.3. Los diferentes números de discos a utilizar pretenden desvelar alguna posible variación en el rendimiento en ambos RAID.

- **Stripe**: es el tamaño en KB de las partes en las que se divide el dato a almacenar en el *array* de discos. Este valor puede afectar a la cantidad de discos utilizados para escribir/leer un dato y por lo tanto a la paralelización de las operaciones de E/S.

Ha sido necesario realizar el despliegue de hardware y software básico. El hardware desplegado ha sido el servidor de almacenamiento y discos especificado en la sección 1.3.1. Todo ello montado en el armario también descrito en la misma sección.

A continuación se muestran unas fotografías del despliegue hardware realizado:

Ilustración 36: despliegue hardware (1).

Los discos se introducen en el chasis del servidor utilizando las bahías disponibles para ello, las cuales permiten añadir y extraer discos al servidor sin necesidad de apagarlo.

Ilustración 37: despliegue hardware (2).

El servidor, marcado en rojo, ha sido introducido en un hueco libre del armario junto con el resto de servidores en producción. La instalación no ha interferido con otros servidores que han podido continuar funcionando normalmente.

4.3.2.3. Sistema operativo y kernel

El sistema operativo instalado es GNU/Linux Ubuntu 10.04 LTS, durante la instalación se han aplicado las configuraciones por defecto. Se ha elegido esta distribución por su soporte a largo plazo en actualizaciones y por ser específica para

servidores. Dichas configuraciones por defecto incluyen los siguientes valores [34] que afectan al funcionamiento de la *page cache* presentada en la sección 2.2.2:

- **dirty_writeback_centisecs** (500 centisegundos): frecuencia con la que se despierta *pdflush* para escribir páginas *dirty* expiradas en disco.

- **dirty_expire_centiseconds** (3000 centisegundos): páginas *dirty* que superen este tiempo se consideran expiradas.

- **dirty_background_ratio** (10%): porcentaje de memoria total del sistema ocupado con páginas *dirty*, a partir del cual *pdflush* comenzará a escribir dichas páginas en disco aunque no estén expiradas.

- **dirty_ratio** (20%): porcentaje de memoria total del sistema ocupado con páginas *dirty*, a partir del cual un proceso que está generando más páginas *dirty* se ve obligado a sincronizar sus escrituras en disco.

4.3.2.4. Sistema de ficheros

El sistema de ficheros, a utilizar durante estas pruebas, será el encargado de almacenar los datos generados por la herramienta IOzone. Es una parte indispensable del sistema, ya que gestiona a nivel de byte las operaciones de escritura y lectura realizadas.

Sistemas de ficheros como FAT32 o ext2, presentados anteriormente en la sección 2.2.2, se han descartado debido a su antigüedad y a que, debido a sus características, no satisfacen los requisitos de los usuarios de unas aulas GNU/Linux. Sus limitaciones son principalmente en tamaño máximo de archivo, rendimiento y robustez ante errores en otras capas del sistema.

El sistema de ficheros ext3 podría ser un candidato, ya que sigue siendo ampliamente usado en sistemas en producción estables. En cambio, distribuciones GNU/Linux como Debian, Ubuntu o CentOS (Red Hat) basan todo su sistema en la versión posterior, ext4, por lo que existe un gran apoyo por parte de las comunidades de software libre para este sistema de ficheros. Dichas comunidades están formadas en su mayor parte por desarrolladores de empresas punteras como Intel o IBM.

En cuanto a XFS, es un sistema de ficheros que por sus características cumple con las necesidades de este proyecto. Supera en uso a ext4 en sistemas donde los requisitos de espacio son muy exigentes.

Finalmente, para la realización de las pruebas se ha decidido utilizar el sistema de ficheros ext4. Además de por las razones comentadas, por las siguientes razones:

- Cumple con las necesidades teóricas del proyecto.
- Su predecesor, ext3, ya fue utilizado por las aulas GNU/Linux de la URJC con éxito por lo que resulta familiar en este tipo de entornos.

4.3.2.5. Tamaño de bloque del sistema de ficheros

El tamaño de bloque del sistema de ficheros es la unidad mínima que maneja el sistema de ficheros en las operaciones de escritura y lectura. El tamaño se especifica en bytes, la elección de dicho tamaño está relacionada con el uso dado al sistema de ficheros, como indica la página de manual del comando *mkfs.ext4* [35].

No se han aplicado optimizaciones importantes en esta fase del proyecto por lo que se ha aplicado el tamaño de bloque por defecto que es de 4096 bytes.

4.3.2.6. Tamaño de stride y stripe-width

Como se ha explicado en la sección 2.2.1, en algunos RAID el dato a escribir es dividido en fragmentos de menor tamaño, cada fragmento es enviado a un disco del *array* propiciando escritura en paralelo y aumentando el rendimiento.

El tamaño de *stride* es el número de bloques del sistema de ficheros que completan cada fragmento del dato. Este valor es utilizado durante la creación del sistema de ficheros para que los metadatos se repartan por cada disco de forma equitativa propiciando un mayor rendimiento [35].

En cambio, *stripe-width* es el número de bloques del sistema de ficheros que serían necesarios para crear un fragmento para todos y cada uno de los discos del RAID. Se debe tener en cuenta el número de discos que forman el RAID y el tipo de RAID utilizado. Este valor es utilizado para evitar cálculos innecesarios del valor de paridad en RAIDs que lo requieran como RAID 5 [35], aumentando así el rendimiento.

4.3.2.7. Ejecución del estudio

A continuación, se detalla cada uno de los pasos seguidos para realizar las pruebas para cada una de las configuraciones de RAID propuestas.

- **Configuración del RAID**: consiste en preparar la configuración de la controladora RAID en función de los tipos de RAID propuestos. En la sección 1 del Anexo B - Procedimientos se pueden observar los pasos realizados para aplicar la configuración.

- **Creación del sistema de ficheros**: consiste en preparar el espacio de almacenamiento disponible, gracias al nuevo RAID, de una estructura lógica para albergar datos. La creación del sistema de ficheros se ha realizado a medida de las características del RAID, configurando adecuadamente *stride* y *stripe-width*. Ver sección 2 del Anexo B - Procedimientos.

- **Ejecución de IOzone**: consiste en ejecutar la herramienta de benchmarking sobre el sistema de ficheros recién creado. La herramienta genera información que es volcada a un fichero de texto, dicho fichero de texto se ha escrito sobre el sistema de ficheros del sistema operativo para que dichas escrituras interfieran lo menos posible en la ejecución de la prueba. Cada ejecución de este paso ha requerido del orden de entre 12 y 24 horas para terminar. Ver sección 3.1 del Anexo B - Procedimientos.

A continuación, se eligió la configuración con mejor rendimiento obtenido de las pruebas anteriores y sobre dicha configuración se aplican los siguientes pasos:

- **Estudiar carga del servidor de almacenamiento**: consiste en leer el fichero */proc/diskstats* durante la ejecución de IOzone, al igual que se hizo en el servidor de aulas durante su uso, para obtener datos de operaciones de E/S realizadas y poder compararlas con el sistema actual de aulas.

- **Analizar efectos del uso de NFS**: consiste en utilizar IOzone en el cliente NFS para comprobar el rendimiento de un sistema de ficheros montado por red.

4.4. Resultados

En esta sección se muestran los resultados obtenidos para los dos estudios realizados. En el caso de los resultados obtenidos con IOzone, para las operaciones de E/S más comunes, solo se incluyen aquellas más relevantes.

4.4.1. Resultados del comportamiento de aulas GNU/Linux

En esta sección se muestran los resultados de los dos estudios realizados para las aulas ya existentes.

4.4.1.1. Tamaño de ficheros

La siguiente imagen muestra una gráfica de tipo *boxplot*, representa un conjunto de datos mediante sus cuartiles. Dicho conjunto de datos es el tamaño de cada fichero del directorio */home* en el sistema actual de aulas GNU/Linux de la URJC.

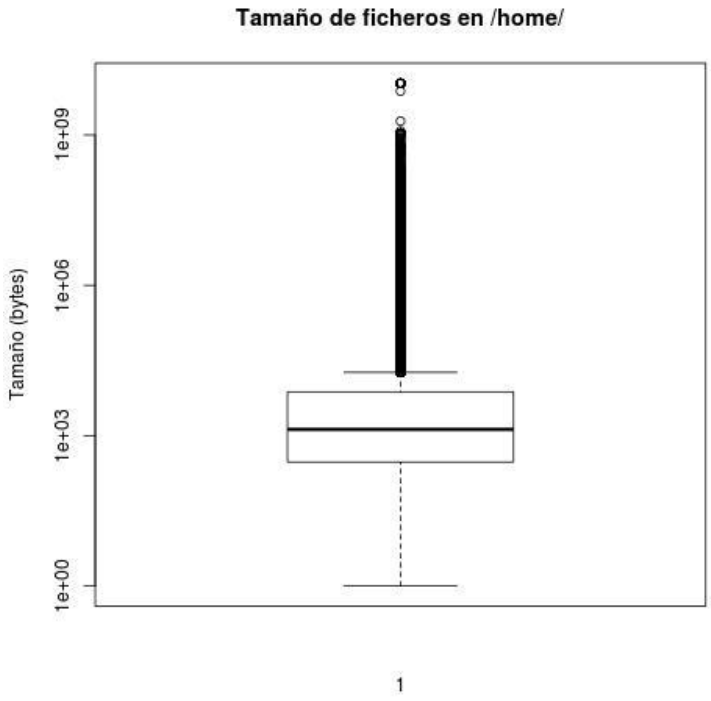

Ilustración 38: boxplot de tamaño de ficheros en /home.

A continuación, la leyenda utilizada para explicar la gráfica:

- **Q1**: cuartil 1, es la mediana de la primera mitad de datos de la serie.
- **Q2**: cuartil 2, es la mediana de todos los datos de la serie.
- **Q3**: cuartil 3, es la mediana de la segunda mitad de datos de la serie.
- **RIC**: Rango Inter Cuartílico, es equivalente a Q3 – Q1.

El *boxplot* anterior tiene las siguientes características:

- El borde inferior de la caja representa el Q1.
- La línea horizontal en el interior de la caja representa el Q2.
- El borde superior de la caja representa el Q3.
- La longitud del bigote superior representa el valor de Q3 + (RIC*1,5).
- La longitud del bigote inferior alcanza el valor de Q1 – (RIC*1,5).
- Cualquier valor fuera del rango que marcan los bigotes es dibujado con un círculo vacío y son considerados valores extraños u *outliers*.

A continuación, información cuantitativa sobre los ficheros estudiados:

- El tamaño de la muestra es de 4.088.153 ficheros que ocupan un total aproximado en el servidor de almacenamiento de la URJC de 356 GB.
- El fichero de menor tamaño es de 1 byte.
- Q1 tiene un valor de 299 bytes.
- Q2 tiene un valor de 1.228 bytes.
- Q3 tiene un valor de 7.528 bytes.
- La media tiene un valor de 2.232.000 bytes.
- El fichero de mayor tamaño es de 10.740.047.872 bytes (~10 GB).

- La desviación estándar tiene un valor de 53.030.146 bytes.

4.4.1.2. *Carga del servidor durante su uso*

Las siguientes imágenes muestran, de forma acumulativa, las escrituras y lecturas realizadas en el servidor de almacenamiento de las aulas GNU/Linux de la URJC. Se han recogido datos durante un periodo de tiempo de 48 horas aproximadamente, desde las 9:18 jueves 15 de noviembre de 2012 hasta las 11:38 sábado 17 de noviembre de 2012. Nótese que todas las horas referidas en esta sección, incluido las de las gráficas, están en GMT.

Gráfica de escrituras y lecturas enfrentadas:

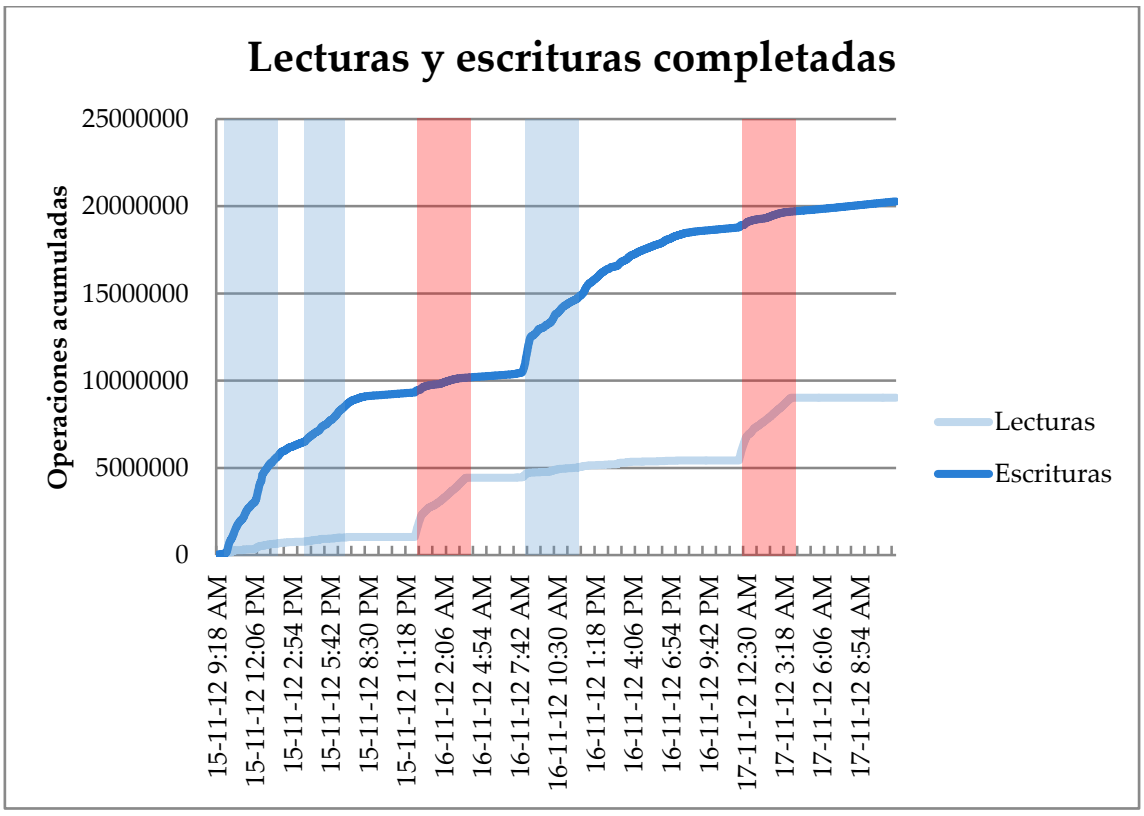

Ilustración 39: lecturas y escrituras completadas.

El eje Y muestra el número de operaciones acumuladas por el dispositivo RAID del servidor de aulas. El eje X muestra la fecha, hora y minuto en la que se realizó la

medición del fichero */proc/diskstats*. En azul claro las lecturas y en azul oscuro las escrituras.

Además, los horarios en los que hubo clase en las aulas aparecen marcados con una franja transparente de color azul.

El sistema de aulas tiene un sistema de *backups* que se ejecuta cada noche a partir de las 11:00 PM, dicho sistema incrementa el número de lecturas drásticamente en el servidor durante un periodo de tiempo de 2 horas aproximadamente, dicho periodo de tiempo está marcado con una franja transparente de color rojo.

Gráfica de lecturas completadas en detalle:

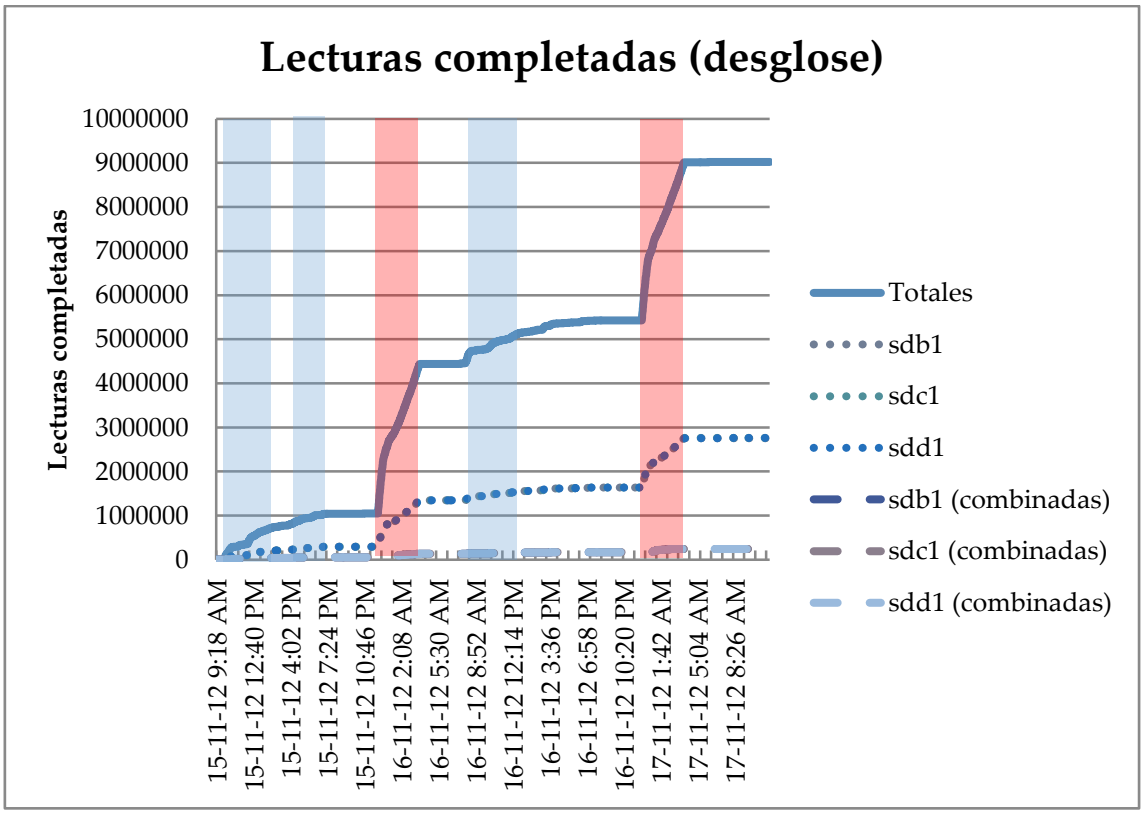

Ilustración 40: lecturas completadas (desglose).

El eje Y muestra el número de operaciones de lectura acumuladas. El eje X muestra la fecha, hora y minuto en la que se realizó la medición del fichero */proc/diskstats*. En

línea continua las operaciones de lectura totales producidas en el dispositivo RAID del servidor de aulas.

En línea de puntos, el desglose de lectura producidas en cada disco que forma el RAID y en línea discontinua las lecturas combinadas, ver sección 4.3.1.2. Nótese que el número de operaciones acumuladas en cada disco es prácticamente idéntico, además se producen más lecturas normales por disco que lecturas combinadas.

Además, los horarios en los que hubo clase en las aulas aparecen marcados con una franja transparente de color azul. Como se comenta en la gráfica anterior, el sistema de *backups* incrementa el número de lecturas durante su ejecución, se refleja mediante una franja transparente de color rojo.

Gráfica de escrituras completadas en detalle:

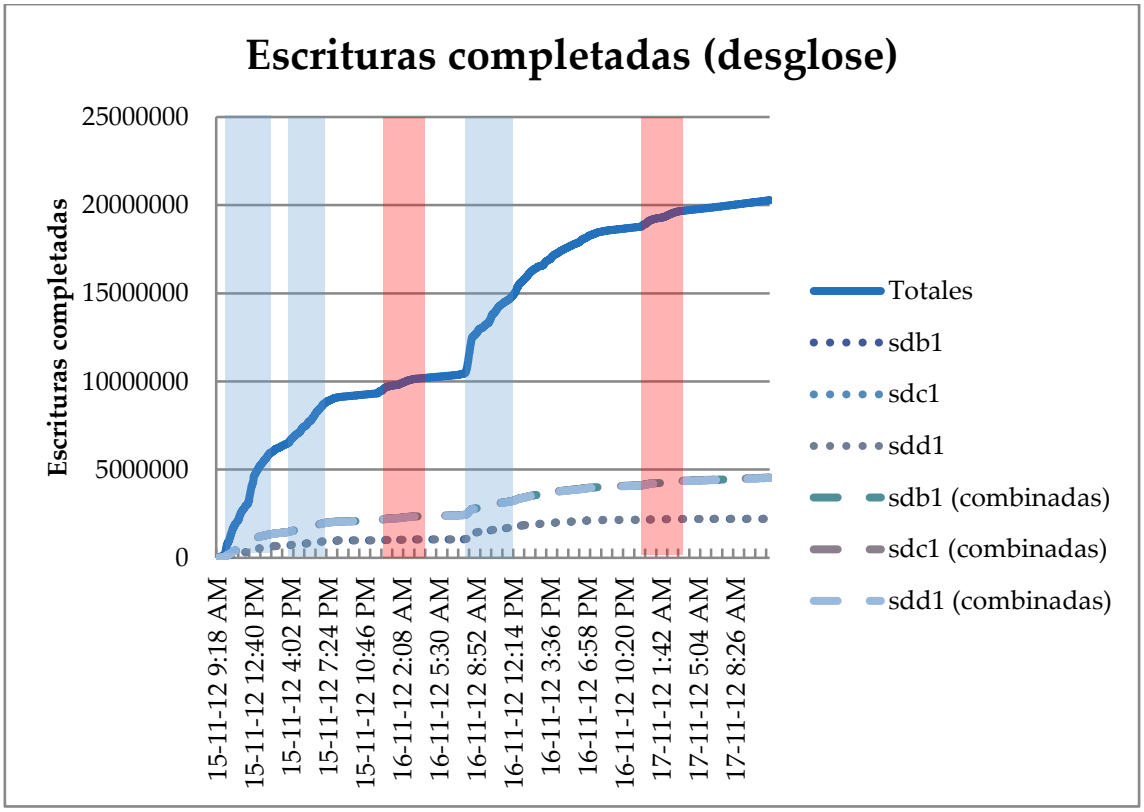

Ilustración 41: escrituras completadas (desglose).

El eje Y muestra el número de operaciones de escritura acumuladas. El eje X muestra la fecha, hora y minuto en la que se realizó la medición del fichero */proc/diskstats*. En línea continua las operaciones de escritura totales producidas en el dispositivo RAID del servidor de aulas.

En línea de puntos el desglose de escrituras producidas en cada disco que forma el RAID y en línea discontinua las escrituras combinadas, ver sección 4.3.1.2. Nótese que el número de operaciones acumuladas en cada disco es prácticamente idéntico, además se producen más escrituras combinadas por disco que escrituras normales.

Además, los horarios en los que hubo clase en las aulas aparecen marcados con una franja transparente de color azul. Como se comenta en la primera gráfica de esta sección con más detalle, el sistema de *backups* incrementa el número de lecturas durante su ejecución, se refleja mediante una franja transparente de color rojo.

4.4.2.Resultados del comportamiento de las tecnologías estudiadas

En esta sección se muestran los resultados del estudio de comportamiento de las tecnologías de almacenamiento contempladas.

4.4.2.1. Operaciones frecuentes

Como se ha explicado al inicio de la sección 4.4, únicamente se incluyen las gráficas más significativas. Para ello, se han establecido las operaciones de entrada y salida más frecuentemente realizadas de todas aquellas que la herramienta IOzone soporta, ver sección 4.3.2.1.

Durante el uso de los terminales en un aula, los alumnos hacen uso de los mismos bajo un régimen de usuarios de escritorio. En estas sesiones los usuarios trabajan en la realización de prácticas, apertura de aplicaciones y consulta de información en Internet.

La realización de prácticas en aulas GNU/Linux en su mayoría requieren el uso de un editor de texto, los cuales no realizan tarea exhaustiva de entrada y salida. Habitualmente abren ficheros ya existentes, cargándolos en memoria para que el usuario pueda modificarlos y posteriormente salvarlos de nuevo en disco. Estos ficheros pueden ocupar un tamaño desde unos pocos bytes, en caso de ficheros de texto plano, o hasta MB, en caso de ficheros correspondientes a paquetes ofimáticos.

La apertura de aplicaciones requiere carga de ficheros binarios en memoria, habitualmente estos ficheros ocupan KB o MB y requieren accesos de lectura secuencial durante la carga de la aplicación.

Por último, la consulta de información en Internet requiere el uso de navegadores web. Dichas aplicaciones pueden llegar a ser bastante pesadas durante el cacheo de información. Navegadores como Google Chrome mantienen una cache que requiere el mantenimiento de numerosos ficheros, dichos ficheros suelen tener un tamaño aproximado de decenas de KB donde leen y escriben continuamente.

Por lo tanto, por esto motivos y teniendo en cuenta las definiciones de la sección 4.3.2.1, se considera que las operaciones más utilizadas son *re-write* y *re-read*.

Las gráficas presentadas en esta sección tienen en común algunas características que se explican a continuación:

- Se identifican claramente tres mesetas. Esto es debido a las caches, la primera meseta representa los mejores rendimientos obtenidos para ficheros que caben en la cache de la CPU, la segunda meseta representa rendimientos más modestos para ficheros de mayor tamaño que solo caben en la memoria o también llamada *page cache*, ver sección 2.2.2, y, finalmente, la tercera meseta que muestra los peores rendimientos para ficheros mayores que la *page cache* que obligan a un acceso al disco físico.

- Algunas zonas de las gráficas aparecen completamente planas con un rendimiento de 0 KB por segundo. Esto es debido a que algunas combinaciones no son probadas por IOzone, como por ejemplo, realizar transferencias de bloques de datos de 1024 KB en ficheros de tamaño menor o transferencias de muy pocos KB en ficheros muy grandes, que alargarían demasiado la prueba en el tiempo.

Operación re-write

A continuación, las gráficas para operaciones de tipo *re-write*, tanto para la configuración con mejor rendimiento de RAID 5 como de RAID 10.

- RAID 5 sobre 12 discos, *stripe* 128 KB:

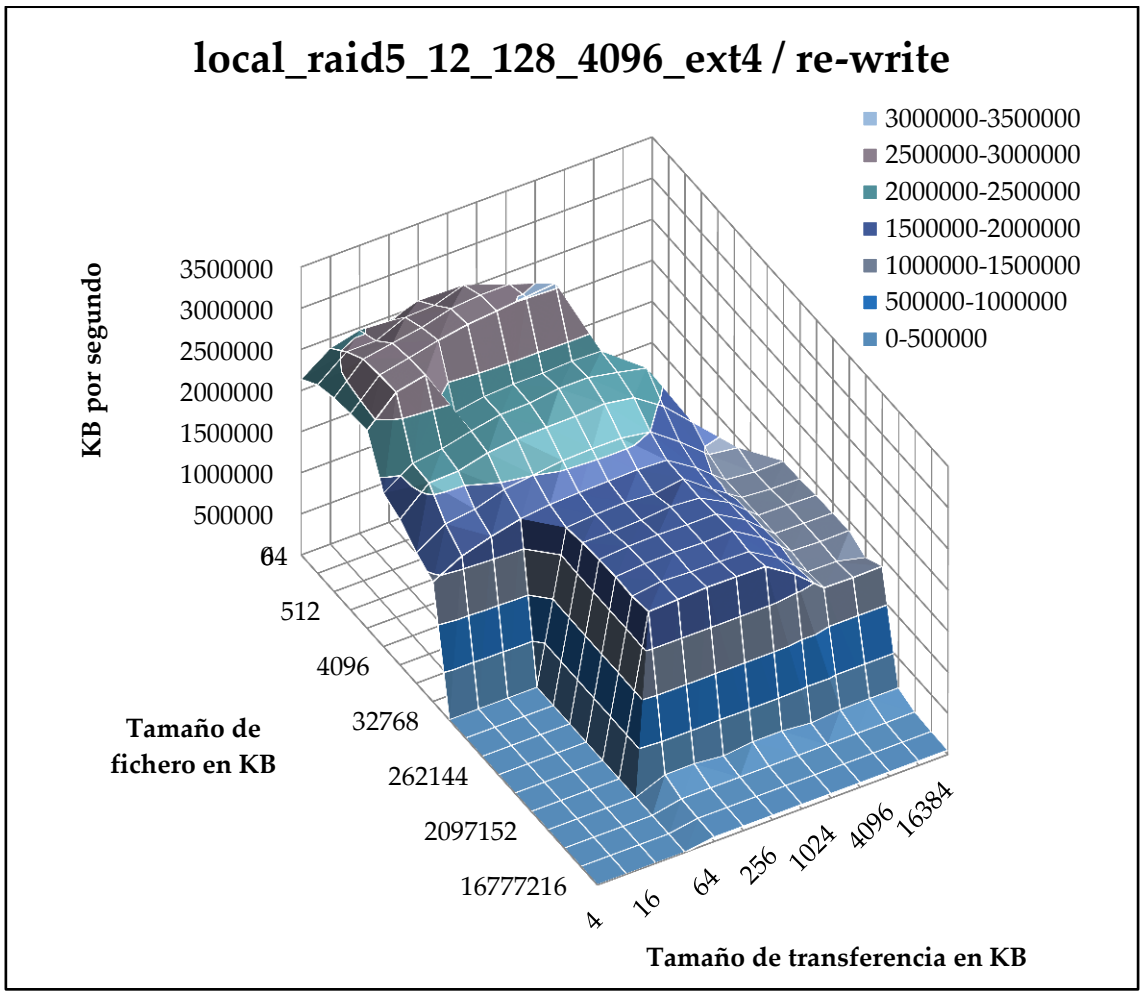

Ilustración 42: local_raid5_12_128_4096_ext4 / re-write.

En esta gráfica se observan rendimientos entorno a los 2,5 y 3,5 GB por segundo para ficheros menores de 512 KB.

Además, para ficheros de más de 512 KB hasta un total de 1 GB, se obtienen rendimientos que van desde 1,7 GB hasta algo más de 2,5 GB por segundo.

Para tamaños de fichero de más de 2 GB, el rendimiento cae drásticamente hasta obtener valores asociados al acceso al dispositivo físico o disco duro.

Finalmente, cuando el tamaño de la transferencia supera los 4096 KB, se observa una bajada del rendimiento situándolo alrededor de 1,5 GB por segundo.

- RAID 10 sobre 12 discos, *stripe* 1024 KB:

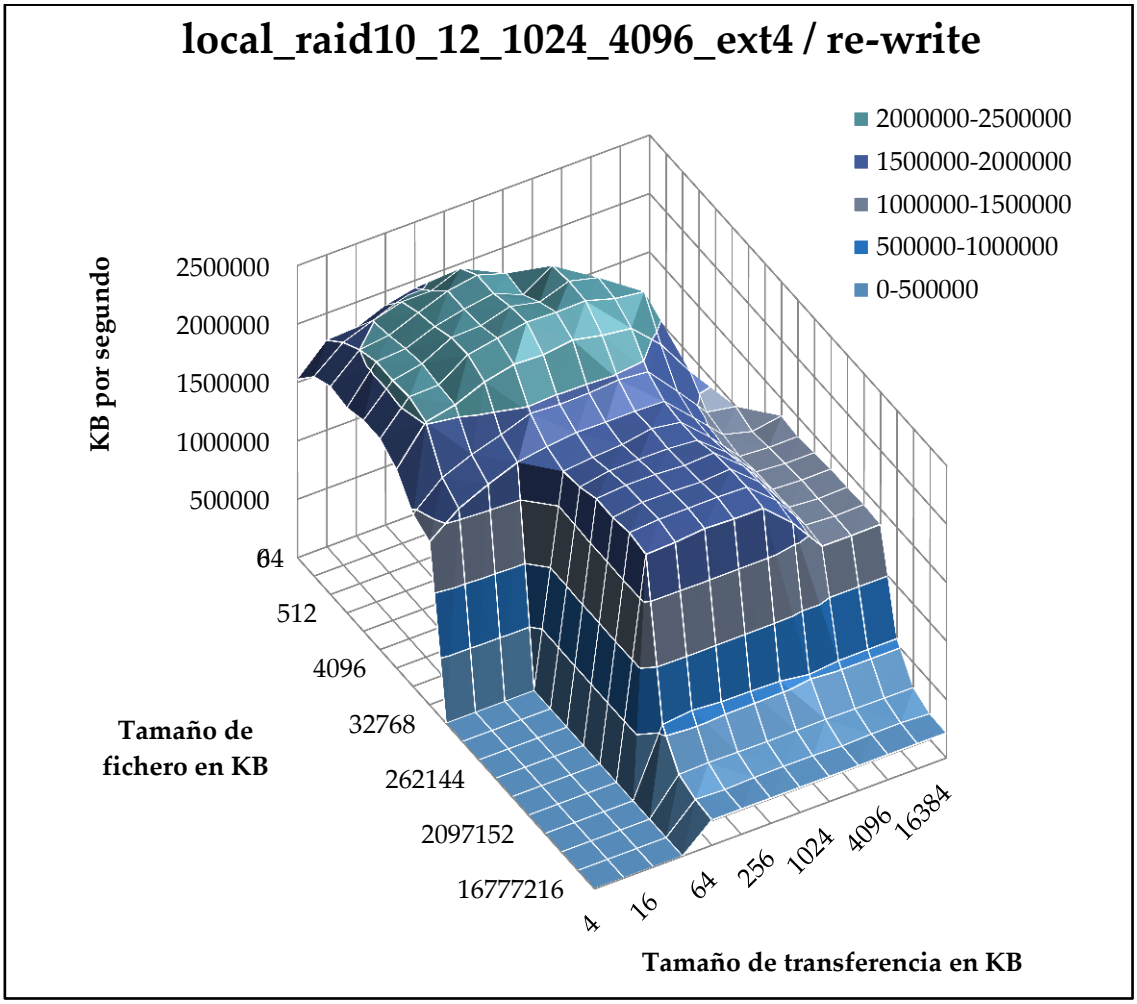

Ilustración 43: local_raid10_12_1024_4096_ext4 / re-write.

En esta gráfica se observan rendimientos entorno a los 2 y 2,5 GB por segundo para ficheros menores de 4096 KB.

Además, para ficheros de más de 4096 KB hasta un total de 1 GB, se obtienen rendimientos que van desde algo más de 1,5 GB hasta los 2 GB por segundo.

Para tamaños de fichero de más de 2 GB, el rendimiento cae drásticamente hasta obtener valores asociados al acceso al dispositivo físico o disco duro.

Finalmente, cuando el tamaño de la transferencia supera los 4096 KB, se observa una bajada del rendimiento situándolo alrededor 1,5 GB por segundo.

Operación re-read

A continuación, las gráficas para operaciones de tipo *re-read*, tanto para la configuración con mejor rendimiento de RAID 5 como de RAID 10.

- RAID 5 sobre 12 discos, *stripe* 128 KB:

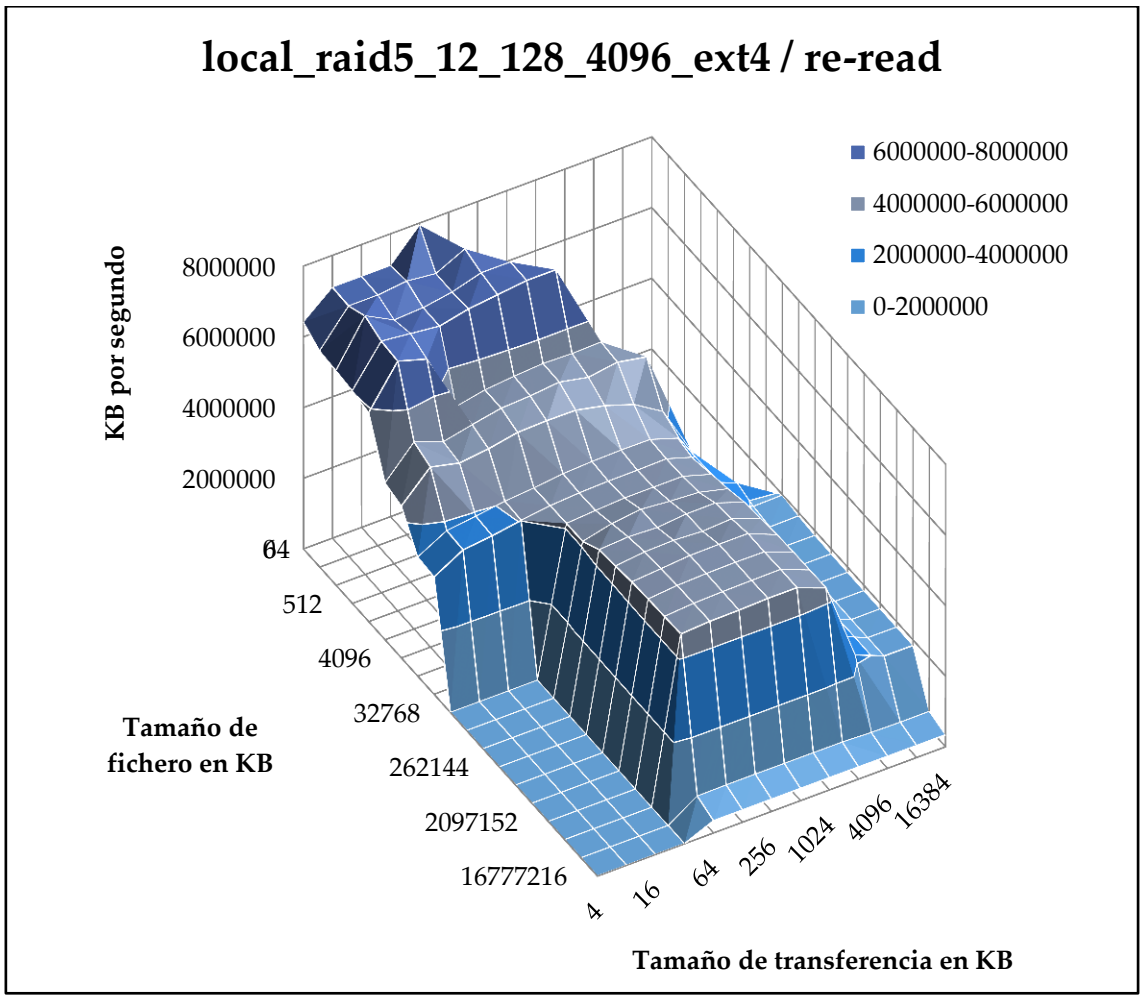

Ilustración 44: local_raid5_12_128_4096_ext4 / re-read.

En esta gráfica se observan rendimientos entorno a los 6 y 8 GB por segundo para ficheros menores de 1024 KB.

Además, para ficheros de más de 1024 KB hasta un total de 4 GB, se obtienen rendimientos que van desde algo más de 4 GB hasta los 6 GB por segundo.

Para tamaños de fichero de más de 8 GB, el rendimiento cae drásticamente hasta obtener valores asociados al acceso al dispositivo físico o disco duro.

Finalmente, cuando el tamaño de la transferencia supera los 4096 KB, se observa una bajada del rendimiento situándolo alrededor 2 GB por segundo.

- RAID 10 sobre 12 discos, *stripe* 1024 KB:

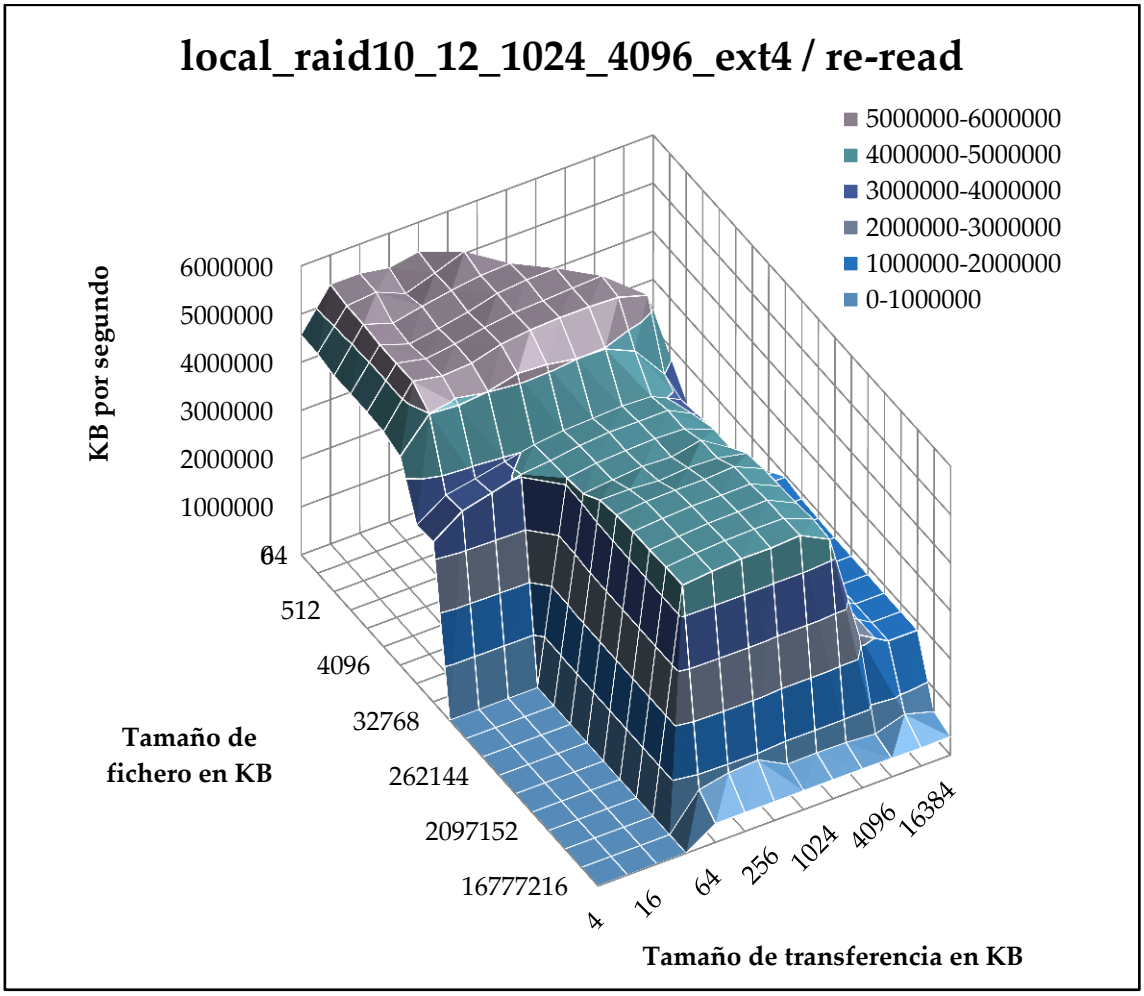

Ilustración 45: local_raid10_12_1024_4096_ext4 / re-read.

En esta gráfica se observan rendimientos entorno a los 4 y 6 GB por segundo para ficheros menores de 4 GB.

Además, para ficheros de más de 1024 KB hasta un total de 4 GB, se obtienen rendimientos que van desde algo más de 4 GB hasta los 6 GB por segundo.

Para tamaños de fichero de más de 8 GB, el rendimiento cae drásticamente hasta obtener valores asociados al acceso al dispositivo físico o disco duro.

Finalmente, cuando el tamaño de la transferencia supera los 4096 KB, se observa una bajada del rendimiento situándolo alrededor 2 GB por segundo.

4.4.2.2. *Carga del servidor de almacenamiento*

A continuación, una gráfica comparativa entre las operaciones de escritura y lectura acumuladas por el servidor de almacenamiento existente de la URJC y el servidor de almacenamiento creado en el presente proyecto ejecutando IOzone localmente.

Como se indica en la sección 4.3.2.7, se ha utilizado la configuración con mejor rendimiento en las anteriores pruebas, RAID 5 sobre 12 discos, *stripe* 128 KB.

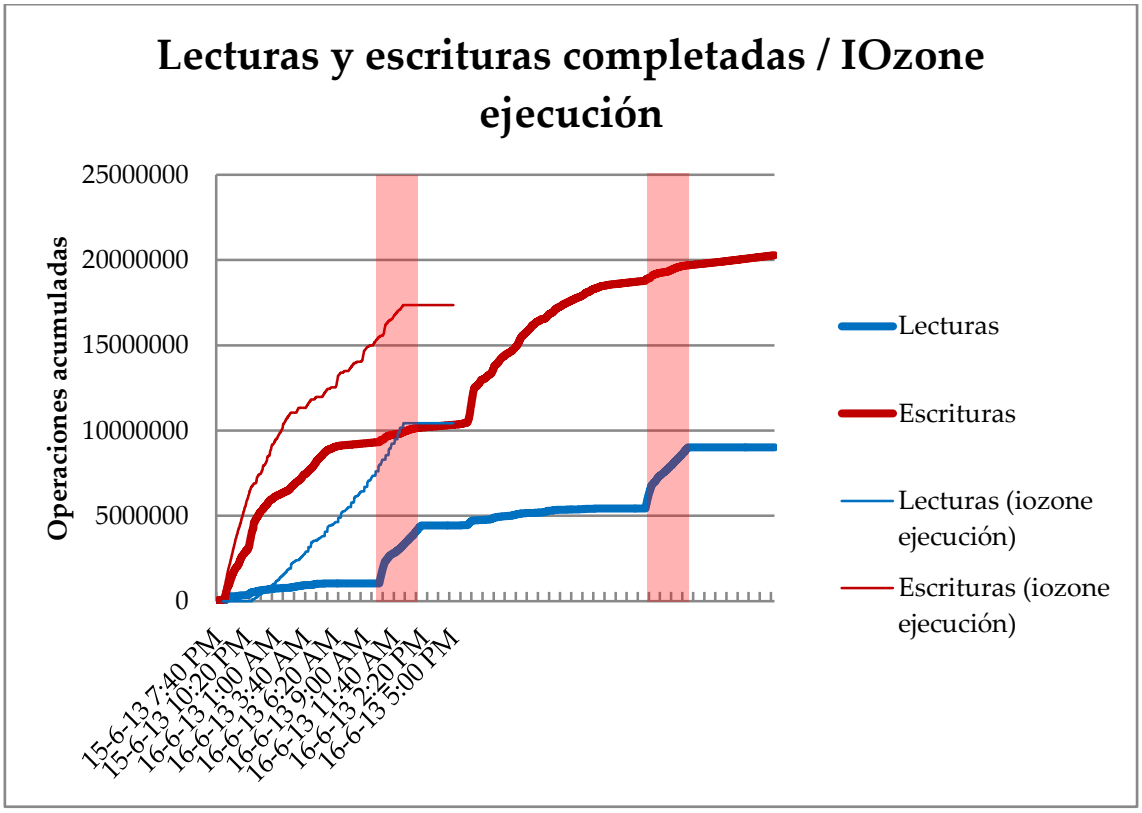

Ilustración 46: lecturas y escrituras completadas / IOzone ejecución.

Las líneas con mayor grosor corresponden al sistema actual de almacenamiento de aulas GNU/Linux de la URJC. Las líneas con menor grosor corresponden al servidor de almacenamiento de este proyecto.

Las muestras de carga del servidor de almacenamiento del proyecto bajo la ejecución de IOzone se han realizado durante un periodo de tiempo aproximado de 24 horas, que es el tiempo que tarda IOzone en completar el *benchmark*. Al contrario que la muestra

de datos de carga del servidor de la URJC que han sido tomadas durante 48 horas como se explica en la sección 4.4.1.2.

El eje X muestra la fecha y hora correspondientes a la recogida de datos de la ejecución de IOzone en el servidor de almacenamiento del proyecto. Los tiempos están ajustados en el eje X, de forma que entre un punto A y otro B del eje X hay el mismo periodo de tiempo para ambas muestras. Los tiempos están en GMT.

Las franjas de tiempo marcadas con un área rojo claro representan el periodo de tiempo en el que se ejecutaron los *backups* en el sistema de la URJC.

La pendiente de cada línea muestra el número de operaciones realizadas en un periodo tiempo determinado.

No deben tenerse en cuenta las pendientes de lecturas de la línea azul gruesa durante las franjas de tiempo dónde se realizan los *backups*, ya que corresponden a lecturas realizadas por el sistema de *backups* en el servidor de la URJC.

4.4.2.3. Efectos del uso de NFS

La utilización de NFS como sistema de ficheros por red puede causar problemas de rendimiento en los clientes debido a la alta latencia de la red, también llamado RTT. Dicha latencia es alrededor de 300-400 milisegundos.

Por este motivo, se ha hecho un estudio de las latencias ofrecidas sobre el sistema de almacenamiento, siendo utilizado por un cliente de las aulas GNU/Linux de la URJC con 4 GB de memoria RAM. Dicho cliente utiliza NFS para acceder al sistema de ficheros exportado por el servidor. La configuración de *page cache* en el sistema cliente es aquella proporcionada por defecto por la instalación del sistema operativo, similar a la utilizada en el servidor de almacenamiento y explicada en la sección 4.3.2.3.

A continuación, se presentan los resultados para los dos tipos de operaciones ya presentadas en este documento.

Operación re-write

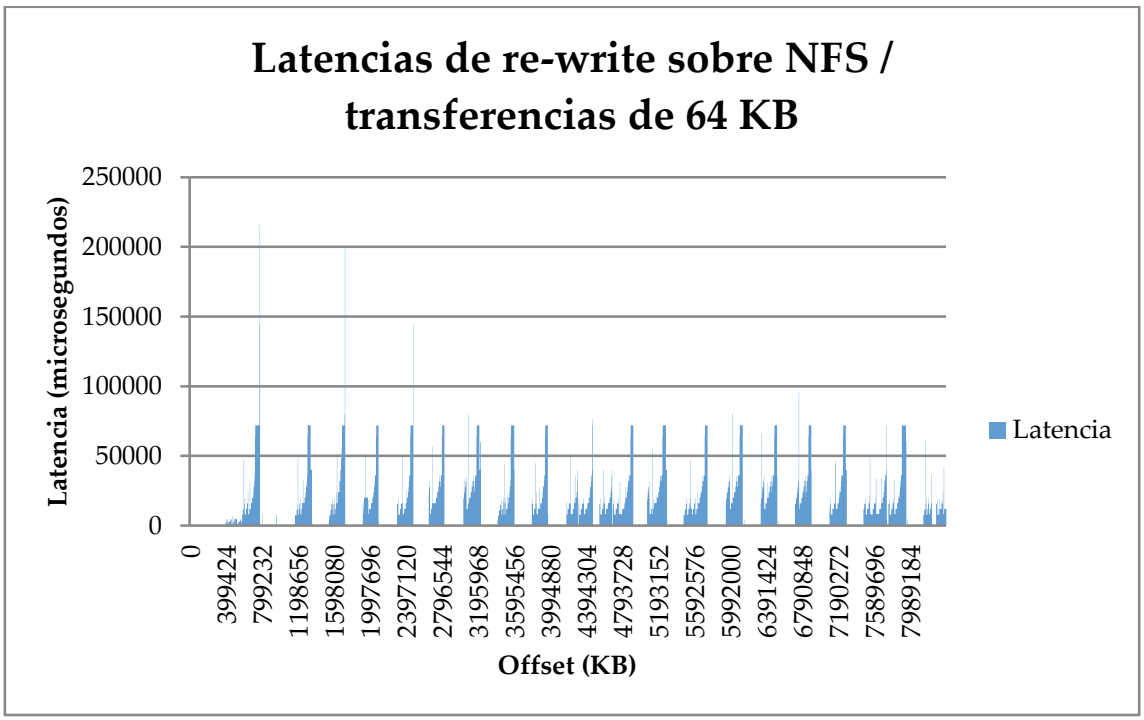

Ilustración 47: latencias de re-write sobre NFS.

En esta gráfica se observan latencias para operaciones re-write, de 64 KB, que en su mayoría son inferiores a los 75.000 microsegundos. Salvo alguna medida puntual, todas están dentro de las latencias propias que proporcionan las caches del sistema cliente.

La serie sigue un ciclo, el cual se repite con mayor frecuencia según aumenta el tamaño del fichero. Esto es debido a que, según se utilizan ficheros lo suficientemente grandes, el sistema llena sus caches. Esto provoca que, para poder escribir un dato más, antes tenga que liberar algunas de las páginas en estado *dirty*, y esto produce un retraso en las operaciones de escritura.

Operación re-read

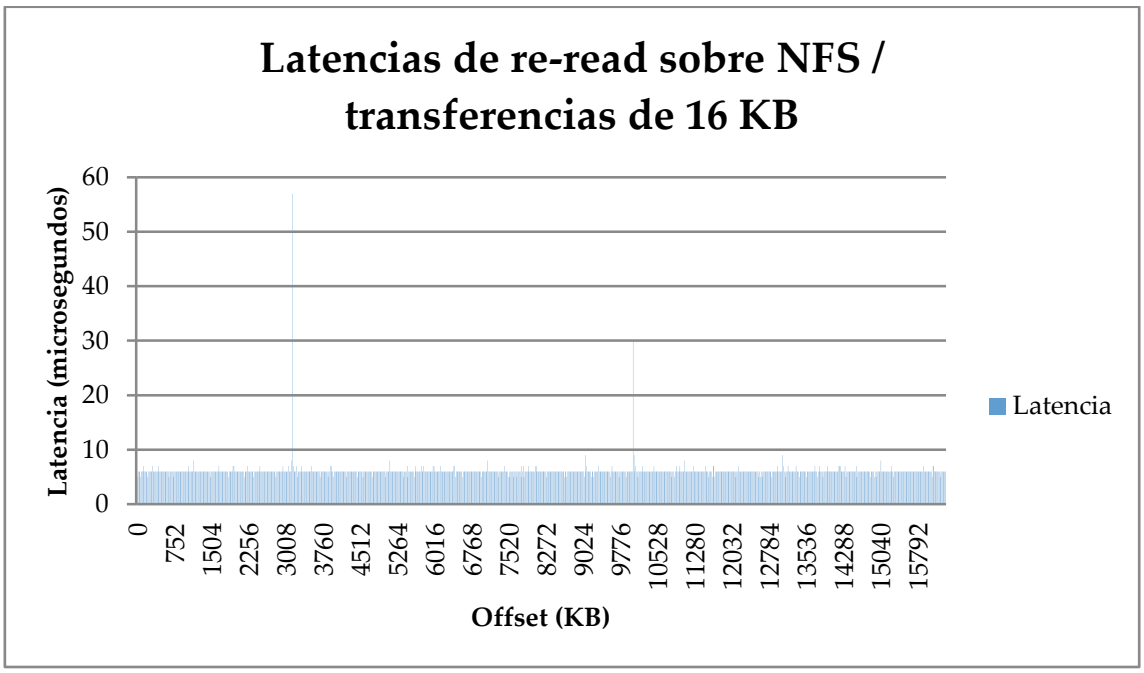

Ilustración 48: latencias de re-read sobre NFS.

En esta gráfica se observan latencias generalizadas para operaciones re-read, de 16 KB. Dichas latencias están alrededor de los 6-7 microsegundos sobre *offsets* en ficheros desde 0 KB hasta aproximadamente 15 MB. Se observan un par de medidas extremas que rondan los 30 y 57 microsegundos respectivamente.

Ninguna de las medidas presentadas se acerca a la latencia ofrecida por la red, sino que se asemejan a las latencias esperables para operaciones sobre un sistema de ficheros local. El motivo por el cual sucede esto es la existencia de las ya mencionadas caches también en el sistema cliente.

4.5. Conclusiones

En esta sección se presentan algunas conclusiones extraídas de los resultados anteriores y que se han tenido en cuenta en el diseño propuesto en la sección 5.

Durante la realización de los estudios se han encontrado parámetros de configuración no conocidos a priori que, a pesar de que su configuración por defecto ha generado resultados satisfactorios, han sido objeto de modificación en la fase de diseño.

4.5.1. Clientes GNU/Linux

El estudio de tamaños de ficheros de los usuarios de las ya existentes aulas GNU/Linux indica que la mayor parte de los ficheros son pequeños. El 75% de los ficheros no superan aproximadamente los 8 KB de tamaño, teniendo en cuenta este dato y el contenido habitual de las prácticas de los alumnos, el cual se caracteriza por ser tareas relacionadas con la programación o edición de documentos, se concluye que el sistema debe ser eficiente manejando especialmente ficheros de estos tamaños, por lo que la utilización de un tamaño de bloque de 4 KB sería adecuado.

Debido al tamaño de más de la mitad de los ficheros, que no superan los 4 KB, utilizar este tamaño de bloque provoca pequeñas pérdidas de espacio para ficheros menores de dicho tamaño. En cambio, mejorará el rendimiento porque bastará con leer o escribir un bloque para leer o escribir un fichero. En cuanto a ficheros mayores de 4 KB, serán necesario menos operaciones que si se usara un tamaño de bloque menor.

En cuanto al uso de ficheros de gran tamaño pertenecientes a máquinas virtuales usadas por los alumnos, es posible preparar arquitecturas de virtualización que permiten, al servidor de almacenamiento, cachear los pequeños ficheros que hay en el interior de dichas VM agilizando, de este modo, sus operaciones de entrada y salida. Estas arquitecturas evitan el uso de ficheros imagen.

En cuanto a la carga generada por las aulas en el sistema de almacenamiento actual, se concluye que el sistema resultante de este proyecto debe ser capaz de soportar, al menos, cargas de la misma intensidad que el sistema actual de la URJC. Para ello se deben observar las pendientes de las gráficas de carga de la sección 4.3.1.2.

4.5.2. Servidor de almacenamiento

Los resultados obtenidos en las pruebas directas sobre el servidor de almacenamiento permiten concluir que:

- El RAID 5 proporciona mejor rendimiento que el RAID 10 a pesar de estar sujeto al cálculo de paridad durante las escrituras. Esto es debido al uso de más discos para paralelizar operaciones que el RAID 10. Además, proporciona una mayor capacidad de almacenamiento. En cambio RAID 10 sigue proporcionando una mayor robustez ante el fallo de discos.

- El rendimiento obtenido para ficheros pequeños es especialmente bueno en comparación con aquellos que ocupan mayor tamaño, gracias a los mecanismos estudiados de *page cache* con su configuración por defecto. Esto se adapta a las necesidades de uso.

- Con la mejor configuración aplicada según los resultados, se ha conseguido generar una carga mayor o igual que la generada en el sistema actual de la URJC, por lo que el nuevo sistema será capaz de soportar, al menos, la carga generada por las aulas de la URJC. Además de proporcionar otra serie de ventajas relacionadas con aspectos como tolerancia a fallos o disponibilidad.

4.5.3. Uso de NFS

Los resultados del impacto de la capa de NFS, que permite a clientes utilizar el sistema de almacenamiento remoto, arrojan que el usuario no debería notar que accede a ficheros mediante la red en la mayoría de los casos.

Como se ha explicado, el sistema de *page cache* en los clientes permitirá realizar operaciones sobre ficheros de forma inmediata en la mayoría de casos previstos, lo que se traducirá en una experiencia de usuario más agradable y completamente transparente con ficheros de tamaño pequeño y medio.

En cambio, es posible que, en transferencias de información de varios megas sobre ficheros no cacheados, el usuario note cierta lentitud debido al ancho de banda limitado de la red.

Capítulo 5

5. Diseño y configuración

Este capítulo recoge la cuarta fase del proyecto, muestra el detalle del diseño del sistema en todas sus capas o niveles y cómo dicho diseño se ha aplicado en la práctica mediante procedimientos.

En la primera sección se muestra un diagrama general por capas, en las siguientes secciones se especifica en detalle cada una de las capas y su diseño.

5.1. Diagrama de componentes

En esta sección se muestra un diagrama de componentes del sistema. El sistema tiene tres capas sobre las cuales se apoya todo el proceso de entrada y salida de datos para su almacenamiento. Las capas superiores son aquellas más alejadas de la consolidación del dato en un soporte físico y las inferiores son aquellas más cercanas a los subsistemas físicos que soportan todo el conjunto del sistema.

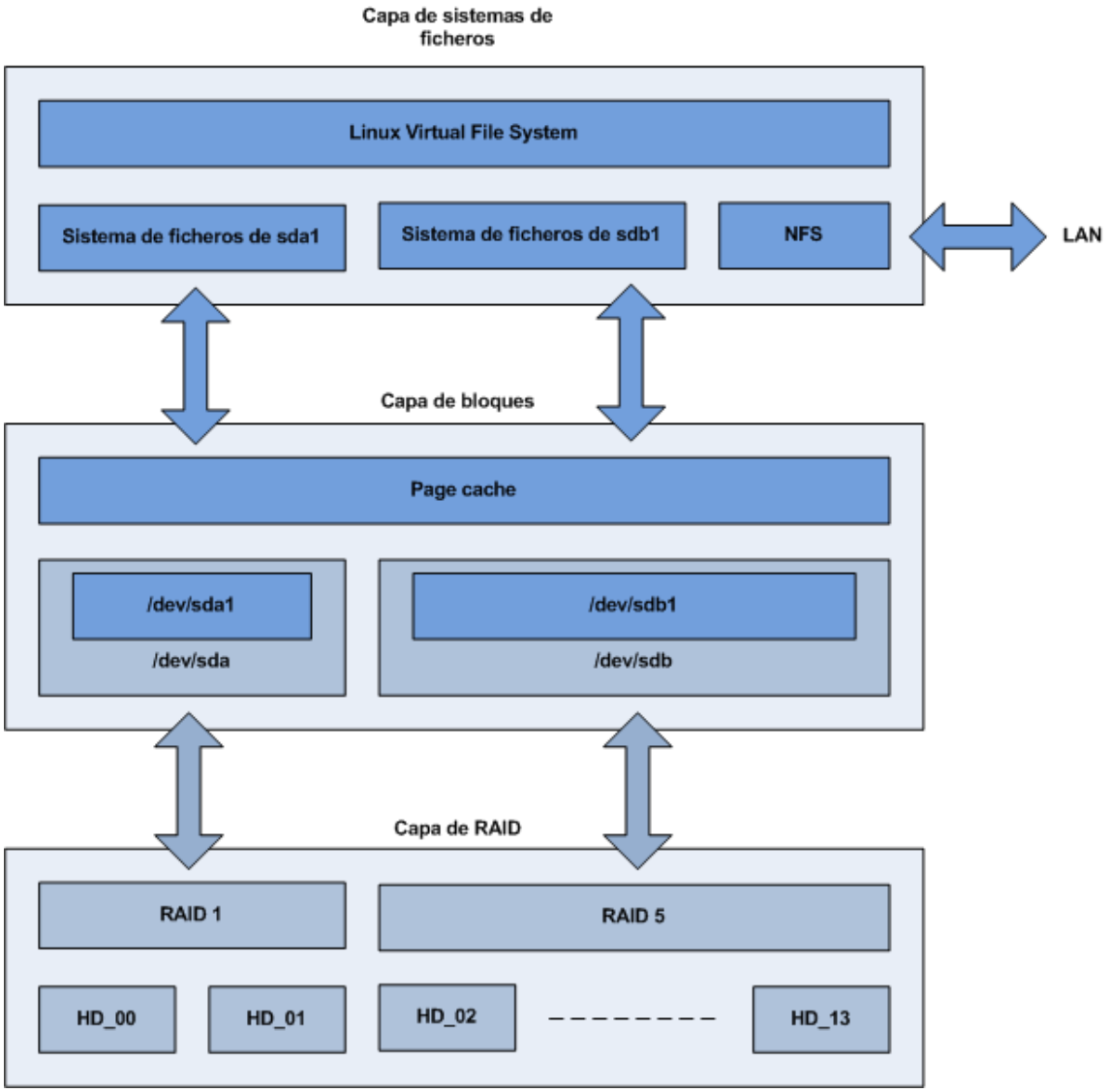

Ilustración 49: diagrama de componentes.

En cuanto a la comunicación de datos, las flechas bidireccionales muestran la comunicación entre subcomponentes de distintas capas. Dentro de cada capa, los subcomponentes en niveles superiores son abstracciones más generales de los que tienen inmediatamente debajo. Además, solo existe comunicación entre subcomponentes del mismo color que están en diferentes niveles de abstracción.

En la capa de RAID se muestra una vista general de cómo se han utilizado los discos físicos para la creación de los RAID. Concretamente se puede observar lo siguiente:

- Cada disco físico está representado mediante los nombres HD_XX, siendo XX un número de disco.

- Se ha creado un RAID 1 utilizando los discos HD_00 y HD_01.
- Se ha creado un RAID 5 utilizando los discos desde HD_02 hasta HD_13.

En la capa de bloques se muestra los dispositivos de bloques necesarios sobre los que se realizán las operaciones para consolidar los datos. Se puede observar los siguientes detalles:

- Existe un dispositivo de bloques llamado */dev/sda*, que representa el RAID 1 creado en la capa inferior.
- Existe un dispositivo de bloques llamado */dev/sdb*, que representa el RAID 5 creado en la capa inferior.
- Cada dispositivo de bloques tiene una partición representada respectivamente mediante los dispositivos de bloques */dev/sda1* y */dev/sdb1*.
- Ambas particiones se comunican con el sistema *page cache*, explicado en la sección 2.2.2.

En la capa de sistemas de ficheros se muestran los sistemas de ficheros asociados a cada partición y el subcomponente *Linux Virtual File System*, ver sección 2.2.5. En ella se pueden ver los siguientes detalles:

- Cada dispositivo de bloques estará gestionado por un sistema de ficheros.
- NFS es un sistema de ficheros capaz de recibir peticiones y de enviar respuestas por red utilizando un protocolo específico.
- *Linux Virtual File System* es la abstracción que permite mostrar un único árbol de directorios en el sistema. Las peticiones de NFS sobre la parte del árbol exportada se atienden a través de dicha abstracción.

En las siguientes secciones se detalla la configuración necesaria en cada componente y subcomponente para la realización del sistema de almacenamiento.

5.2. Capa de RAID

En esta sección se detalla el diseño de la capa de RAIDs. Concretamente se especifican las configuraciones necesarias en la creación de los dos RAID que sustentan el almacenamiento secundario del servidor.

La asignación de discos a cada RAID es secuencial, eligiendo siempre el primer disco sin asignar disponible en la máquina.

5.2.1.RAID 1

El primer RAID se destina al almacenamiento del propio sistema operativo y herramientas del servidor de almacenamiento. Este RAID únicamente tiene que soportar las operaciones que genere el propio sistema operativo por lo que no necesita una configuración especial de alto rendimiento.

El motivo por el cual se elige un RAID de tipo 1 es porque añade seguridad ante la posible pérdida de uno de los discos, de modo que el sistema continuará funcionando sin afectar al servicio de los propios clientes ni al funcionamiento del servidor.

El RAID ha sido configurado siguiendo los siguientes parámetros:

Parámetro	Valor	Notas
Tipo	RAID 1	
Número de discos	2	
Nombre	OS	
Read cache	Activada	Mejora el rendimiento en operación de lectura sobre datos de reciente uso.
Write cache	Desactivada	Operaciones de escritura se dan por realizadas cuando han sido consolidadas en disco.

Tabla 26: configuración de RAID 1.

A pesar de que, como se ve más adelante en la sección 5.4, se utiliza un sistema asíncrono de operaciones en la capa de sistemas de ficheros, se considera que aquellas

operaciones a realizar directamente sobre el dispositivo RAID por parte del *kernel* Linux, deben ser confirmadas únicamente cuando han sido consolidadas en disco. De esta forma, el sistema será conocedor de si finalmente un dato fue finalmente escrito o no sin necesidad de mantener bloqueada la llamada del usuario a la espera de una respuesta.

La configuración de este RAID se ha realizado siguiendo el procedimiento 1 del Anexo B - Procedimientos.

5.2.2.RAID 5

El segundo RAID se destina al almacenamiento de los datos de los clientes GNU/Linux. Este RAID soporta las operaciones generadas por todos los clientes utilizando el sistema de ficheros exportado por NFS, recibirá mucha carga.

El motivo por el cual se elige RAID de tipo 5 es porque permite almacenar los datos repartidos entre varios discos, esto mejora enormemente el rendimiento de las operaciones de lectura y escritura porque se realizan en paralelo. A cambio, el RAID debe calcular la paridad que, si bien retrasa las operaciones de escritura, se compensa si el número de discos es grande. Además, permite la pérdida de uno de los discos sin que se pierda información almacenada y sin tener que detener el sistema.

El RAID ha sido configurado siguiendo los siguientes parámetros:

Parámetro	Valor	Notas
Tipo	RAID 5	
Número de discos	12	
Nombre	STORAGE00	
Stripe size	128 KB	Ver sección 4.4.2.1. Los resultados arrojan que los mejores rendimientos se obtuvieron para este tamaño de stripe.
Read cache	Activada	Mejora el rendimiento en operación de lectura sobre datos de reciente uso.
Write cache	Desactivada	Operaciones de escritura se dan por realizadas cuando han sido consolidadas en disco.

Tabla 27: configuración de RAID 5.

Al igual que con el RAID 1 anterior, como se ve más adelante en la sección 5.4 y a pesar de que se utiliza un sistema asíncrono de operaciones en la capa de sistemas de ficheros, se considera que aquellas operaciones a realizar directamente sobre el dispositivo RAID por parte del *kernel* Linux deben ser confirmadas únicamente cuando han sido consolidadas en disco. De esta forma, el sistema será conocedor de si finalmente un dato fue finalmente escrito o no sin necesidad de mantener bloqueada la llamada del usuario a la espera de una respuesta.

La configuración de este RAID se ha realizado siguiendo el procedimiento 1 del Anexo B - Procedimientos.

5.3. Capa de bloques

En esta sección se detalla el diseño de la capa de bloques. Concretamente se especifican las configuraciones necesarias en la creación de las particiones que dan

lugar a los dispositivos de bloque */dev/sda1* y */dev/sdb1*, además, también se especifica la configuración a aplicar para el funcionamiento de *page cache*.

5.3.1.Particiones

La partición */dev/sda1* se crea durante la instalación del sistema operativo, debe ocupar la totalidad del dispositivo de bloques que identifica el RAID 1, es decir, */dev/sda*. Durante el proceso de instalación se pide indicar información relativa al sistema de ficheros a utilizar, esto se detalla la sección 5.4.

La partición */dev/sdb1* es la encargada de almacenar toda la información de los terminales GNU/Linux, es creada en el dispositivo de bloques que identifica al RAID 5, es decir, */dev/sdb*. Dicha partición debe ser creada una vez el sistema operativo está instalado siguiendo el procedimiento 4 del Anexo B - Procedimientos.

Los detalles de creación para dicha partición son los siguientes:

Parámetro	Valor	Notas
Tabla de particiones	GPT, ver sección 2.1.1.1.	Para soportar particiones de más de 2 TB.
Tipo de partición	Primaria	
Alineación	Óptima, ver sección 2.1.1.	La partición debe estar alineada con los bloques físicos del disco para un mayor rendimiento.
Tamaño	Máximo tamaño permitido por el dispositivo */dev/sdb*.	

Tabla 28: parámetros de configuración de /dev/sdb1.

5.3.2.Page cache

El sistema *page cache* de Linux es configurable para adaptarlo a un tipo de uso específico del sistema de entrada y salida. En función de lo encontrado durante tercera fase, se detallan a continuación los valores aplicados para adaptar *page cache* al caso específico de un servidor de almacenamiento [36].

Parámetro	Valor	Notas
dirty_background_ratio	5	5% de la memoria libre + memoria usable.
dirty_ratio	60	60% de la memoria libre + memoria usable.

Tabla 29: parámetros de configuración para page cache.

De este modo, el demonio *pdflush* empezará a escribir páginas en estado *dirty* cuando estas superen el 5% de la memoria libre + usable. Es interesante que este valor sea bajo, para escribir cuanto antes en disco los datos que están a la espera, de este modo se reduce el riesgo de pérdida de datos si se produce un apagado o reinicio no ordenado del servidor y, además, se descarga la memoria dejando páginas libres para reutilizar si el *kernel* lo considera oportuno.

Además, el valor a partir del cual el sistema fuerza a que todas las escrituras sean síncronas para que dejen de crearse páginas en estado *dirty* es 60%. Está muy por encima del calor por defecto y permite utilizar la memoria del servidor para aumentar la cache disponible, como contrapartida, si se alcanza el valor de 60% el rendimiento se verá reducido drásticamente ya que los procesos no podrán realizar escrituras asíncronas y tendrán que esperar a que los datos sean consolidados en disco. Esta situación perdura hasta que el % de páginas *dirty* en memoria baje del umbral establecido.

No se espera que dicho umbral se alcance con facilidad ya que, gracias al primer parámetro, *pdflush* trabaja desde el comienzo para que así sea y, además, todos los datos que puedan llevar al sistema a esta situación provienen de la red, la cual es más lenta que el RAID 5 consolidando datos.

Finalmente, para aplicar dicha configuración se ha aplicado el procedimiento definido en la sección 5 del Anexo B - Procedimientos.

5.4. Capa de sistemas de ficheros

En esta sección se detalla el diseño de la capa de sistemas de ficheros. Concretamente se especifican las configuraciones necesarias en la creación del sistema de ficheros de la partición */dev/sda1*, de la partición */dev/sdb1* así como las configuraciones necesarias para la exportación del sistema de ficheros de */dev/sdb1* mediante NFS.

5.4.1.Sistema de ficheros de /dev/sda1

El sistema de ficheros para el dispositivo de bloques */dev/sda1* alberga los datos del propio sistema del servidor de almacenamiento. Dicho sistema de ficheros tiene las siguientes características:

Parámetro	Valor	Notas
Sistema de ficheros	ext4	Ver sección 4.3.2.4.
Punto de montaje	/	
Tamaño de bloque	4096 bytes	Configuración por defecto.

Tabla 30: sistema de ficheros para /dev/sda1.

Cualquier otra posible característica a configurar se ha mantenido en su valor por defecto.

5.4.2.Sistema de ficheros de /dev/sdb1

El sistema de ficheros para el dispositivo de bloques */dev/sdb1* alberga los datos de los clientes GNU/Linux de las aulas. Dicho sistema de ficheros tiene las siguientes características:

Parámetro	Valor	Notas
Sistema de ficheros	ext4	Ver sección 4.3.2.4.
Tamaño de bloque	4096 bytes	Ver sección 4.5.1.
Stride	32	Ver sección 2 del Anexo B - Procedimientos.
Stripe-width	352	Ver sección 2 del Anexo B - Procedimientos.

Tabla 31: sistema de ficheros para /dev/sdb1.

Además, el sistema ha sido configurado para que conozca algunos parámetros que le permitan gestionar dicho sistema de ficheros de forma adecuada. Dichos parámetros son los siguientes:

- **Punto de montaje**: */data/*. Especifica el punto del árbol de directorios dónde debe montarse dicho sistema de ficheros.
- **rw**: se permite leer y escribir en el sistema de ficheros.
- **suid**: se permite el uso de los bit suid y sgid.
- **dev**: se interpretan los dispositivos de bloque en el sistema de ficheros.
- **exec**: se permite la ejecución de programas almacenados en el sistema de ficheros.
- **auto**: se monta el sistema de ficheros en el punto de montaje especificado de forma automática en el arranque del sistema.
- **nouser**: no se permite que un usuario normal monte o desmonte el sistema de ficheros.
- **async**: las operaciones de entrada y salida sobre el sistema de ficheros se realizan de forma asíncrona.
- **noatime**: no se actualizan los tiempos de acceso a inodos en cada acceso.
- **errors=remount-ro**: si se produce un error en el sistema de ficheros se revoca el permiso de escritura en el mismo de forma automática.
- **dump**: 0, no es necesario realizar *backups* del sistema de ficheros.
- **pass**: 2, prioridad baja al chequeo del sistema de ficheros durante el arranque del sistema. La prioridad alta debe tenerla el sistema de ficheros de */dev/sda1* por ser el que almacena el sistema GNU/Linux del servidor.

Dichas especificaciones se han aplicado siguiendo el procedimiento 6 del Anexo B - Procedimientos.

5.4.3.NFS

El sistema de ficheros NFS se encarga de exportar a la red el sistema de ficheros del dispositivo de bloques */dev/sdb1*. Para realizar la exportación es necesario conocer el valor de los parámetros que definen qué se exporta y bajo qué condiciones.

A continuación, los parámetros bajo los cuales el sistema exporta por red:

- **Hosts clientes**: 193.147.49.0/24. Se ha especificado de forma explícita la red a la que pertenecen los terminales GNU/Linux de la URJC. Existen otras formas más estrictas de identificar qué hosts podrán usar el sistema de ficheros exportado, pero, dado que uno de los objetivos es causar el menor impacto en los terminales, se ha considerado esta alternativa como más adecuada para el problema concreto.

- **rw**: se permiten lecturas y escrituras por parte de los clientes sobre el sistema exportado.

- **async**: las operaciones de escritura se confirman sin esperar a que se consoliden en almacenamiento secundario. Esto incrementa notablemente el rendimiento aunque provoca la posibilidad de pérdida de información si el servidor sufre un apagado o reinicio no controlado.

- **no_subtree_check**: el sistema de almacenamiento no comprueba la localización de un fichero accedido. El manual de NFS recomienda esta configuración en sistema para *homes* de usuario, habitualmente accedidos por usuarios que escriben y cambian nombres de ficheros de forma simultánea.

- **auth_nlm**: con esta opción se obliga a requerir credenciales UID/GID para las peticiones de bloqueo sobre ficheros.

- **mp**: no se exporta el sistema de ficheros si no fue correctamente montado en */data/* previamente, como se indica en la sección 5.4.2. De este modo, se evita que accidentalmente algún cliente pudiera realizar operaciones sobre el directorio */data/* del servidor, en el cual no estaría montado */dev/sdb1*, que recaerían sobre */dev/sda1* con el consiguiente peligro para el propio sistema del servidor de almacenamiento.

- **root_squash**: todas las peticiones del usuario con UID 0, *root*, en el cliente se mapean al usuario anónimo en el servidor. De este modo, en el caso hipotético de que un usuario consiguiera acceso privilegiado en alguno de los

clientes, no podría realizar operaciones sobre el sistema de ficheros remoto con permisos de *root*.

- **Threads**: 16. Se aplica un total de 8 hilos por *core*. Por defecto NFS viene con 4 hilos por *core*. Si el número de clientes aumenta puede que sea necesario aumentar este valor. Dado que se va a manejar un número elevado de clientes simultáneamente, se ha duplicado el valor por defecto. Podría ser necesario aumentarlo si en el fichero */proc/net/rpc/nfsd* se observa que el número de veces que se han utilizado todos los hilos es similar, o ligeramente inferior, al número de veces que se utilizan solo el 50% de los mismos, para ello basta consultar la fila *th* del fichero.

Las presentes especificaciones se han aplicado siguiendo el procedimiento 7 del Anexo B - Procedimientos.

5.5. Red

En esta sección se muestra el diseño de la red de aulas de la URJC, en la cual se ha integrado el servidor de almacenamiento para comprobar que, efectivamente, puede ser utilizado por un entorno de aulas GNU/Linux ya existente.

En la siguiente imagen se muestra la red, representada por un canal cilíndrico azul, y las aulas, representadas por sendos rectángulos azules.

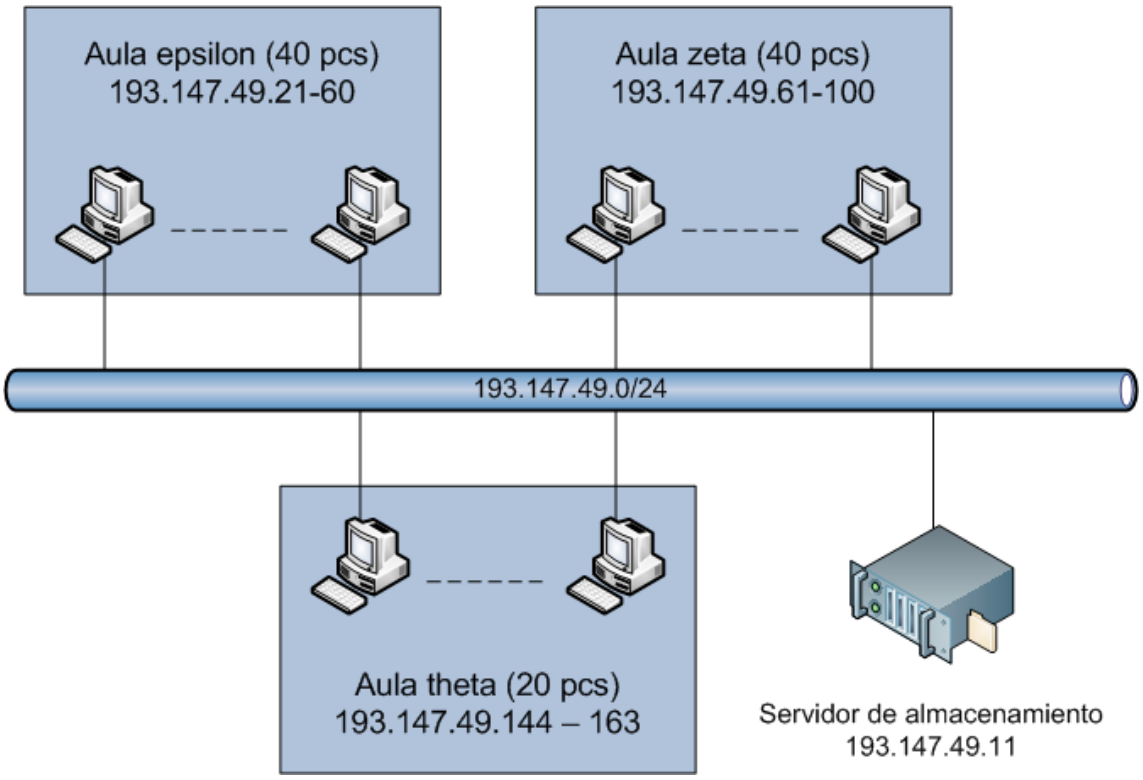

Ilustración 50: diseño de red de aulas GNU/Linux.

Como se puede observar, solo existe una subred que es utilizada por todas las aulas que hacen uso del servidor de almacenamiento. Dado que este proyecto solo abarca el diseño del servidor de almacenamiento y su integración en aulas GNU/Linux ya existentes, no se detallará más la red en este documento.

El servidor de almacenamiento creado utiliza la misma dirección IP de red que el servidor actual con el propósito de facilitar su despliegue, evitando, de este modo, modificaciones en los clientes o cambios en sistemas de seguridad de red.

Por lo tanto, los datos de red del servidor de almacenamiento son:

- **IP**: 193.147.49.11
- **Submáscara de red**: 255.255.255.0
- **Gateway**: 193.147.49.1
- **Tecnología de capa física y enlace**: Fast Ethernet (100 Mbit/s)

Capítulo 6

6. Integración y pruebas

Este capítulo recoge la quinta fase del proyecto, se muestra la integración del servidor en un sistema de aulas GNU/Linux ya existente y se detallan las pruebas a realizar para comprobar que el sistema cumple los requisitos especificados.

6.1. Integración

La integración del servidor consiste en conectar el servidor de almacenamiento a la red de las aulas GNU/Linux existentes. Como se dispone de la colaboración de la URJC, se ha decidido integrar el servidor en sus aulas GNU/Linux y configurar los clientes para que utilicen el nuevo servidor. Debido a que se ha trabajado con un entorno de producción real, únicamente ha sido posible realizar la integración para cuatro clientes GNU/Linux del aula sobre los que se han realizado las pruebas de la sección 6.2.

Este proceso se divide en varios pasos que se describen a continuación en orden de aplicación.

Integrar el nuevo servidor en la red aplicando una IP temporal distinta a la del servidor actual, sería sustituida por la indicada en la sección 5.5 si se realizase una integración completa para todos los PCs de las aulas.

Sincronización de datos con el servidor actual, se debe realizar una copia de los datos del servidora actual al nuevo servidor para permitir a los clientes GNU/Linux conservar los datos tras la integración. En esta sincronización es importante mantener los UID/GIDs y permisos, tanto de ficheros como de directorios. Para ello se ha utilizado el comando *rsync* como se puede ver en la sección 8 del Anexo B - Procedimientos.

Montaje del sistema de ficheros NFS en los clientes GNU/Linux. Para ello basta con conectarse a los clientes GNU/Linux y aplicar la siguiente configuración al fichero */etc/fstab* como se indica en la sección 9 del Anexo B - Procedimientos:

- **Servidor**: <ip temporal> aplicada durante la integración

- **Path remoto**: */data/*
- **Punto de montaje local**: */home/*
- **Tipo de sistema de ficheros**: nfs
- **rw**: con esta opción se permite leer y escribir sobre el sistema de ficheros importado del servidor.
- **rsize=8192**: tamaño del buffer de lectura, recomendado en el manual de NFS para un mayor rendimiento. Reducción de *overhead* en la red por cabeceras TCP y disminución del número de peticiones al servidor.
- **wsize=8192**: tamaño del buffer de escritura, recomendado en el manual de NFS para un mayor rendimiento. Reducción de *overhead* en la red por cabeceras TCP y disminución del número de peticiones al servidor.
- **noatime**: a pesar de que el servidor ya lo contempla, se pide explícitamente que los clientes no soliciten actualizar la información de acceso a ficheros en *inodos*.
- **hard**: los programas clientes quedarán bloqueados esperando cuando el servidor falle, cuando el servidor NFS vuelva a estar disponible continuarán con su tarea normalmente.
- **intr**: se permite el uso de señales para detener operaciones de entrada y salida sobre el sistema de ficheros NFS importado.
- **tcp**: las conexiones con el servidor NFS utilizaran TCP, permitiendo así control de congestión, control de flujo y gestión de reenvíos en caso de pérdidas. Dado que en una integración completa habría muchos PCs, la red tendría una carga no despreciable que podría causar problemas si se usa UDP.
- **bg**: si el montaje del sistema de ficheros remoto NFS falla, el sistema devuelve un mensaje de error pero continúa intentando montarlo en *background*.

Puesta en marcha de los clientes GNU/Linux, consiste en reiniciar los equipos cliente para asegurar que utilizan la nueva configuración y que queden a la espera de alumnos que inicien sesión y comiencen a utilizar el sistema.

6.2. Pruebas

Las pruebas se dividen en dos tipos, las pruebas de integración, que validan el sistema y las pruebas de aceptación, que verifican que se cumplen los requisitos especificados.

Cada prueba cumplirá con el formato de la siguiente tabla:

ID	
Nombre	
Descripción	
Procedimiento	
Resultado	
Requisito de capacidad	
Resultado esperado	
Estado	

Tabla 32: ejemplo de prueba.

El significado de los campos de dicha tabla es el siguiente:

- **ID**: identificador único para cada prueba. Las pruebas de integración tendrán un identificador con el formato PRU-IN-XX y las pruebas de aceptación un formato PRU-AC-XX. En ambos formatos, XX es el número de prueba correspondiente.
- **Nombre**: nombre para la prueba.
- **Descripción**: descripción detallada de la prueba.

- **Procedimiento**: explica cómo realizar la prueba paso a paso.
- **Resultado (sólo para las pruebas de integración)**: expone el resultado obtenido en las pruebas de integración.
- **Requisito de capacidad (sólo para pruebas de aceptación)**: indica el requisito de capacidad relacionado con dicha prueba.
- **Resultado esperado (sólo para las pruebas de aceptación)**: expone el resultado esperado tras la realización de la prueba.
- **Estado**: indica si la prueba se ha realizado o no con éxito.

6.2.1.Pruebas de integración

En esta sección se muestran las pruebas de integración:

ID	**PRU-IN-01**
Nombre	Estado de los discos
Descripción	Se comprueba que todos los discos pertenecientes al servidor de almacenamiento estén disponibles.
Procedimiento	▪ El administrador enciende el servidor de almacenamiento. ▪ El administrador pulsa 'CTRL+A' cuando aparece el mensaje 'Adaptec RAID Configuration Utility!'. ▪ El administrador elige la opción 'Disk Utilities'
Resultado	Todos los discos aparecen online asignados a las correspondientes bahías dónde han sido introducidos.
Estado	Prueba realizada con éxito.

Tabla 33: prueba de integración PRU-IN-01.

ID	**PRU-IN-02**
Nombre	Estado de los RAID
Descripción	Se comprueba que todos los RAID estén en un estado óptimo.
Procedimiento	▪ El administrador enciende el servidor de almacenamiento. ▪ El administrador pulsa 'CTRL+A' cuando aparece el mensaje 'Adaptec RAID Configuration Utility!'. ▪ El administrador elige la opción 'Manage Arrays' ▪ El administrador selecciona el array '00 OS RAID 1'. ▪ El administrador pulsa la tecla ESC. ▪ El administrador selecciona el array '01 STORAGE00 RAID 5'.
Resultado	Tras seleccionar ambos arrays el campo 'Array Status' de ambos RAID es OPTIMAL.
Estado	Prueba realizada con éxito.

Tabla 34: prueba de integración PRU-IN-02.

ID	**PRU-IN-03**
Nombre	Montaje de */dev/sdb1*
Descripción	Se comprueba que el servidor tiene acceso al sistema de ficheros con los datos de los usuarios.
Procedimiento	▪ El administrador enciende el servidor de almacenamiento. ▪ El administrador se autentica con el usuario 'root'. ▪ El administrador ejecuta: 'mount'.
Resultado	El dispositivo */dev/sdb1* aparece montado en el punto de montaje */data/* mediante el siguiente: /dev/sdb1 on /data type ext4 (rw)
Estado	Prueba realizada con éxito.

Tabla 35: prueba de integración PRU-IN-03.

ID	**PRU-IN-04**
Nombre	Sistema de ficheros exportado
Descripción	Se comprueba que el servidor está exportando el directorio /data/ a la red de clientes GNU/Linux.
Procedimiento	▪ El administrador enciende el servidor de almacenamiento. ▪ El administrador se autentica con el usuario 'root'. ▪ El administrador ejecuta: 'ping 193.147.49.1 –c 2' ▪ El administrador ejecuta: 'netstat –atpu' ▪ El administrador ejecuta: 'exportfs'
Resultado	▪ Dos ping llegan a la IP 193.147.49.1: PING 193.147.49.1 (193.147.49.1) 56(84) bytes of data. 64 bytes from 193.147.49.1: icmp_seq=1 ttl=64 time=0.917 ms 64 bytes from 193.147.49.1: icmp_seq=2 ttl=64 time=0.705 ms ▪ Servicios de NFS escuchan peticiones: tcp 0 0 *:57719 *:* LISTEN 31865/rpc.mountd tcp 0 0 *:nfs *:* LISTEN - tcp 0 0 *:44009 *:* LISTEN 805/rpc.statd tcp 0 0 *:sunrpc *:* LISTEN 716/portmap ▪ */data* parece como sistema de ficheros exportado: /data 193.147.49.0/24
Estado	Prueba realizada con éxito.

Tabla 36: prueba de integración PRU-IN-04.

6.2.2.Pruebas de aceptación

En esta sección se muestran las pruebas de aceptación:

ID	**PRU-AC-01**
Nombre	Escritura en home
Descripción	Se comprueba que un usuario puede realizar escrituras en su directorio home.
Procedimiento	▪ El usuario inicia sesión en un cliente GNU/Linux. ▪ El usuario ejecuta: 'mkdir ~/directorio_prueba' ▪ El usuario ejecuta: 'echo $?' ▪ El usuario ejecuta: 'cd ~/directorio_prueba' ▪ El usuario ejecuta: 'dd if=/dev/zero of=./test.dd bs=512 count=200000' ▪ El usuario ejecuta: 'echo $?'
Requisito de capacidad	REQ-C-01
Resultado esperado	▪ El primero *echo* devuelve 0 por pantalla. ▪ El segundo *echo* devuelve 0 por pantalla.
Estado	Prueba realizada con éxito.

Tabla 37: prueba de aceptación PRU-AC-01.

ID	**PRU-AC-02**
Nombre	Lectura en home
Descripción	Se comprueba que un usuario puede realizar lecturas en su directorio home.
Procedimiento	▪ El usuario inicia sesión en un cliente GNU/Linux. ▪ El usuario ejecuta: 'less ~/.bashrc' ▪ El usuario ejecuta: 'echo $?'
Requisito de capacidad	REQ-C-02
Resultado esperado	El comando *echo* devuelve 0 por pantalla.
Estado	Prueba realizada con éxito.

Tabla 38: prueba de aceptación PRU-AC-02.

ID	**PRU-AC-03**
Nombre	Software libre
Descripción	Se comprueba que el sistema ha sido creado utilizando productos con licencias de software libre.
Procedimiento	▪ El administrador enciende el servidor de almacenamiento. ▪ El administrador ejecuta: 'cat /etc/apt/sources.list'
Requisito de capacidad	REQ-C-03
Resultado esperado	Durante la configuración del servidor únicamente han estado activas las ramas *main* y *restricted* del repositorio de paquetes de Ubuntu, [37]. deb http://es.archive.ubuntu.com/ubuntu/ lucid main restricted deb-src http://es.archive.ubuntu.com/ubuntu/ lucid main restricted ## Major bug fix updates produced after the final release of the ## distribution. deb http://es.archive.ubuntu.com/ubuntu/ lucid-updates main restricted deb-src http://es.archive.ubuntu.com/ubuntu/ lucid-updates main restricted
Estado	Prueba realizada con éxito.

Tabla 39: prueba de aceptación PRU-AC-03.

ID	**PRU-AC-04**
Nombre	Gestión de sistema de ficheros delegada
Descripción	Se comprueba que el servidor tiene acceso a la administración del sistema de ficheros que contiene los datos de los usuarios.
Procedimiento	▪ El administrador enciende el servidor de almacenamiento. ▪ El administrador se autentica con el usuario 'root'. ▪ El administrador ejecuta: 'parted –l /dev/sdb'.
Requisito de capacidad	REQ-C-04
Resultado esperado	El comando devuelve la siguiente salida donde se observan la partición dónde reside el sistema de ficheros exportado: Model: Adaptec STORAGE00 (scsi) Disk /dev/sdb: 5492GB Sector size (logical/physical): 512B/512B Partition Table: gpt Number Start End Size File system Name Flags 1 1049kB 5492GB 5492GB ext4 primary
Estado	Prueba realizada con éxito.

Tabla 40: prueba de aceptación PRU-AC-04.

ID	PRU-AC-05
Nombre	Integración sencilla
Descripción	Se envía una breve encuesta explicando el funcionamiento básico del sistema a varios administradores con experiencia laboral y se les pregunta si lo consideran sencillo o si es mejorable.
Procedimiento	▪ Enviar el correo electrónico a José Torrado, Alberto Cacharro y Juan José Amor. Ver Anexo A - Resultados de la encuesta de integración.
Requisito de capacidad	REQ-C-05
Resultado esperado	Los administradores encuestados consideran sencilla la forma de integrar el sistema en las aulas GNU/Linux.
Estado	Prueba realizada con éxito.

Tabla 41: prueba de aceptación PRU-AC-05.

ID	**PRU-AC-06**
Nombre	El sistema soporta fallos
Descripción	Se comprueba que el sistema funciona si ocurren algunos fallos conocidos.
Procedimiento	▪ El administrador enciende el servidor de almacenamiento. ▪ El administrador se autentica con el usuario 'root'. ▪ El administrador quita uno de los cables de las fuentes de alimentación. ▪ El administrador quita uno de los discos correspondientes al RAID 1. ▪ El administrador quita uno de los discos correspondientes al RAID 5. ▪ El administrador ejecuta PRU-AC-01. ▪ El administrador ejecuta PRU-AC-02. ▪ El administrador vuelve a conectar el cable de corriente. ▪ El administrador vuelve a introducir el disco correspondiente al RAID 1. ▪ El administrador vuelve a introducir el disco correspondiente al RAID 5.
Requisito de capacidad	REQ-C-06
Resultado esperado	▪ Las pruebas PRU-AC-01 y PRU-AC-02 se hicieron con éxito tras los fallos provocados a propósito. ▪ Tanto el RAID 1 como el RAID 5 comienzan a reconstruirse tras insertar de nuevo los discos.
Estado	Prueba realizada con éxito.

Tabla 42: prueba de aceptación PRU-AC-06.

ID	**PRU-AC-07**
Nombre	Conservación de UIDs, GIDs y permisos.
Descripción	Se comprueba que el servidor mantiene los UIDs, GIDs y permisos de directorios y ficheros pertenecientes a usuarios de terminales GNU/Linux.
Procedimiento	▪ El administrador enciende el servidor de almacenamiento. ▪ El administrador se autentica con el usuario 'root'. ▪ El administrador ejecuta: 'ls –lhn /data/<usuario>'. ▪ El usuario inicia sesión en un cliente GNU/Linux. ▪ El usuario ejecuta: 'ls –lhn ~'.
Requisito de capacidad	REQ-C-07, REQ-C-08 y REQ-C-09.
Resultado esperado	▪ La ejecución del comando *ls* por parte del administrador devuelve: drwx------ 2 1000 1000 4.0K 2012-12-27 20:14 bin drwx------ 2 1000 1000 4.0K 2012-12-27 20:14 Desktop drwx------ 2 1000 1000 4.0K 2012-12-27 20:14 Documents drwx------ 2 1000 1000 4.0K 2012-12-27 20:14 Downloads ▪ La ejecución del comando *ls* por parte del usuario devuelve: drwx------ 2 1000 1000 4.0K 2012-12-27 20:14 bin drwx------ 2 1000 1000 4.0K 2012-12-27 20:14 Desktop drwx------ 2 1000 1000 4.0K 2012-12-27 20:14 Documents drwx------ 2 1000 1000 4.0K 2012-12-27 20:14 Downloads La primera, tercera y cuarta columna de la respuesta del comando en el servidor de almacenamiento son idénticas a las correspondientes en el comando ejecutado en el cliente GNU/Linux.
Estado	Prueba realizada con éxito.

Tabla 43: prueba de aceptación PRU-AC-07.

ID	**PRU-AC-08**
Nombre	Acceso de terceros a datos.
Descripción	Se comprueba que el servidor ofrece una forma alternativa de acceder a los datos sin el uso de aulas GNU/Linux.
Procedimiento	▪ El administrador conecta a otro servidor cualquiera de la red con suficiente espacio para albergar el contenido de los homes. ▪ El administrador ejecuta el comando: 'rsync –avz –e 'ssh –p 20022' root@193.147.49.11:/data/ /<dir_con_espacio>/'
Requisito de capacidad	REQ-C-10
Resultado esperado	Al finalizar la ejecución del comando *rsync*, se ha obtenido una copia de los datos del directorio */data/* del servidor de almacenamiento en el host utilizado para lanzar el procedimiento.
Estado	Prueba realizada con éxito.

Tabla 44: prueba de aceptación PRU-AC-08.

ID	**PRU-AC-09**
Nombre	Aumento de la capacidad de almacenamiento
Descripción	Se comprueba que el servidor permite aumentar el espacio disponible.
Procedimiento	▪ El administrador enciende el servidor de almacenamiento. ▪ El administrador se autentica con el usuario 'root'. ▪ El administrador realiza un backup en otro lugar de los datos almacenados en el RAID 5. ▪ El administrador comprueba el espacio disponible con el comando 'df –h'. ▪ El administrador apaga el servidor de almacenamiento. ▪ El administrador insertar uno o más discos en las bahías disponibles. ▪ El administrador enciende el servidor. ▪ El administrador eliminar el RAID 5 existente y crea uno nuevo siguiendo la sección 1 del Anexo B - Procedimientos. ▪ El administrador enciende el servidor de almacenamiento. ▪ El administrador se autentica con el usuario 'root'. ▪ El administrador ejecuta los procedimientos 4, 6 del Anexo B - Procedimientos. ▪ El administrador vuelca el backup de los datos sobre el nuevo RAID 5.
Requisito de capacidad	REQ-C-11
Resultado esperado	Se comprueba mediante el comando 'df –h' que el espacio disponible es mayor que el que había al ejecutar el cuarto paso del procedimiento de esta prueba. No existe otra forma de aumentar el espacio ya que la controladora RAID no permite la expansión dinámica de arrays de discos.
Estado	Prueba realizada con éxito.

Tabla 45: prueba de aceptación PRU-AC-09.

ID	**PRU-AC-10**
Nombre	Integridad de datos
Descripción	Se comprueba que el servidor realiza pruebas de integridad del sistema de ficheros de forma automática.
Procedimiento	▪ El administrador enciende el servidor de almacenamiento. ▪ El administrador se autentica con el usuario 'root'. ▪ El administrador ejecuta: 'tune2fs –l /dev/sdb1'.
Requisito de capacidad	REQ-C-12
Resultado esperado	Tras la ejecución del comando se observa una salida que contiene la siguiente información: Mount count: 1 **Maximum mount count: 26** Last checked: Fri Jun 28 17:15:06 2013 **Check interval: 5184000 (2 months)** Next check after: Tue Aug 27 17:15:06 2013
Estado	Prueba realizada con éxito.

Tabla 46: prueba de aceptación PRU-AC-10.

Capítulo 7

7. Planificación y presupuesto

En este capítulo se explica la planificación, la cual incluye un listado de tareas y su correspondiente diagrama de Gantt, y el presupuesto siguiendo la plantilla proporcionada por la Universidad Carlos III de Madrid [38].

7.1. Planificación

En esta sección se detalla la planificación, en primer lugar se presenta una tabla con el listado de tareas con fecha de inicio, fecha de fin y su duración bruta sin fines de semana. En segundo lugar, se presenta un diagrama de Gantt que muestra lo anterior de forma visual, dando un contexto de las tareas a lo largo del desarrollo del proyecto.

7.1.1. Tareas

La siguiente tabla incluye las tareas realizadas en este proyecto.

Tarea	Fecha inicio	Fecha fin	Duración bruta (en días)
Documentación	03/09/2012	17/09/2012	11
▪ Investigación de arquitecturas	03/09/2012	13/09/2012	9
▪ Investigación de tecnologías	10/09/2012	17/09/2012	6
Análisis	18/09/2012	03/10/2012	12
▪ Adquisición de conocimientos sobre uso de aulas	18/02/2012	24/09/2012	5
▪ Obtención de requisitos	25/09/2012	03/10/2012	7
Estudio	04/10/2012	11/03/2013	113
▪ Despliegue hardware	04/10/2012	12/10/2012	7
▪ Familiarización con tecnologías	15/10/2012	10/01/2013	64
▪ Diseño de estudios	05/11/2012	12/11/2012	6
▪ Estudio del sistemas de aulas de la URJC	13/11/2012	10/12/2012	20
▪ Estudio de rendimiento de tecnologías	29/11/2012	11/03/2013	73
Diseño y configuración	12/03/2013	28/03/2013	13
▪ Diseño general	12/03/2013	14/03/2013	3
▪ Diseño específico por niveles	15/03/2013	28/03/2013	10
▪ Creación de procedimientos	20/03/2013	28/03/2013	7
Integración y pruebas	29/03/2013	26/04/2013	21
▪ Integración con terminales GNU/Linux	29/03/2013	05/04/2013	6
▪ Diseño y ejecución de pruebas	08/04/2013	26/04/2013	15
Memoria	18/09/2012	28/06/2013	204
▪ Recopilación de información	18/09/2012	11/03/2013	125
▪ Realización de la memoria	04/02/2013	28/06/2013	105

Tabla 47: lista de tareas del proyecto.

7.1.2. Diagrama de Gantt

A continuación, el diagrama de Gantt de las tareas que aparecen en la tabla anterior.

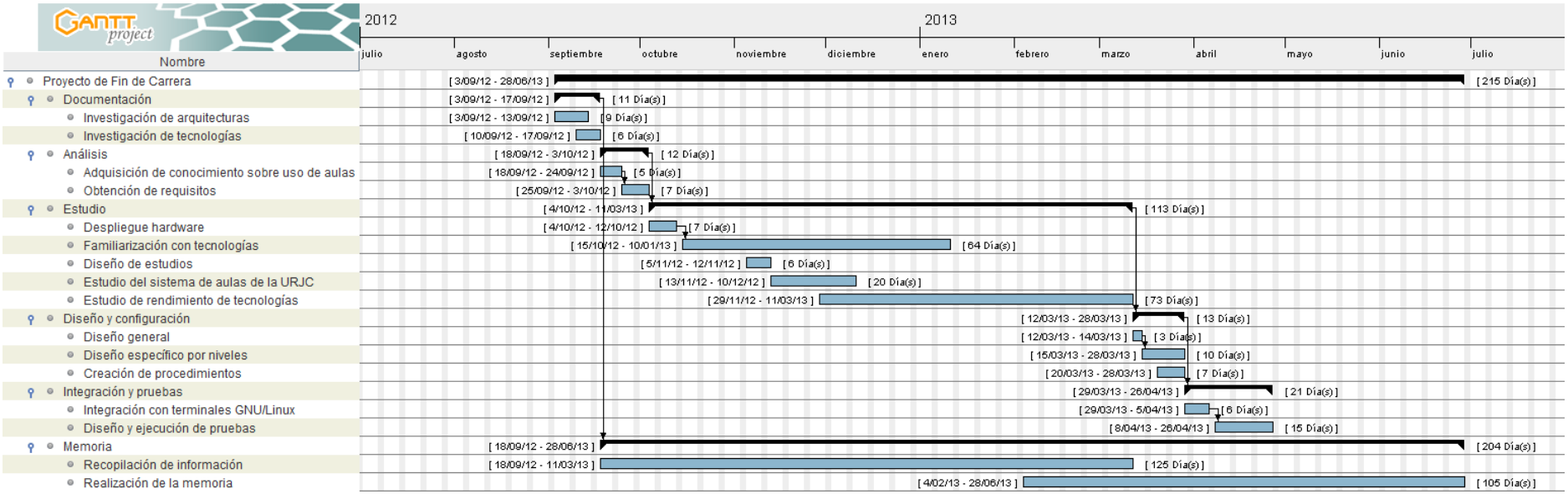

Ilustración 51: diagrama de Gantt.

7.2. Presupuesto

En esta sección se sigue la guía propuesta por la UC3M [39] para presupuestos de proyectos fin de carrera. Se desglosa en las siguientes secciones.

7.2.1. Autor

El autor del presente documento es el ingeniero proyectista Roberto Andradas Izquierdo.

7.2.2. Departamento

Departamento de Informática de la Universidad Carlos III de Madrid.

7.2.3. Descripción del proyecto

El proyecto presenta un sistema a medida de almacenamiento secundario para aulas GNU/Linux.

7.2.4. Presupuesto total del proyecto

El presupuesto total del proyecto asciende a DIECISIETE MIL VEINTE EUROS CON CUARENTA Y CINCO CÉNTIMOS DE EURO.

7.2.5. Desglose presupuestario (costes directos)

En esta sección se desglosa el presupuesto en coste de personal y coste de equipos necesarios para realizar el proyecto.

7.2.5.1. Personal

En el cálculo de los costes de personal del proyecto se han considerado los siguientes aspectos:

- La cantidad de tiempo diaria dedicada al proyecto son 1,5 horas.
- No se contabilizan los siguientes días:

- 24, 25 y 31 de Diciembre de 2012
- 4,5 y 6 de Enero de 2013
- 1 y 2 de Mayo de 2013

La tabla de tareas 7.1.1 resumida sin los días no trabajados queda como sigue:

Tarea	Días reales
Documentación	11
Análisis	12
Estudio	109
Diseño y configuración	13
Integración y pruebas	21
Memoria	198

Tabla 48: días reales trabajados.

Teniendo en cuenta los siguientes datos:

- Total días = 364 días
- Horas al día = 1,5 horas
- Dedicación, 1 hombre/mes = 131,25 horas
- Coste hombre/mes = 2.694,39 €

Mediante la siguiente fórmula se calcula el coste de personal:

$$Coste\ de\ personal\ = \frac{Total\ días\ \times\ Horas\ al\ día}{Dedicación} \times\ Coste\ (hombre/mes)$$

El resultado del coste total de personal del proyecto es ONCE MIL DOSCIENTOS OCHO EUROS CON SESENTA Y SEIS CÉNTIMOS DE EURO.

A continuación, la tabla con el listado del personal que ha intervenido en este proyecto:

Apellidos, nombre	N.I.F	Categoría	Días trabajados	Horas al día	Coste (en €)	Firma
Andradas Izquierdo, Roberto	---	Ingeniero	364	1,5	11.208,66	

Tabla 49: lista de personal del proyecto.

7.2.5.2. Equipos

Para la realización del proyecto se han utilizado los siguientes equipos:

Descripción	Coste sin IVA (en €)	% Uso dedicado al proyecto	Dedicación (meses)	Periodo de depreciación (meses)	Coste imputable (en €)
Equipo de sobremesa	645,50	34	24	60	87,78
Servidor almacenamiento	3.405,14	10	75	60	425,64
Total	4.050,64				513,42

Tabla 50: coste de material del proyecto.

Para el cálculo del coste imputable se ha aplicado la siguiente fórmula:

$$Coste\ imputable\ = \frac{Dedicación}{Periodo\ depreciación} \times Coste\ sin\ IVA\ \times\ \%\ Uso\ dedicado$$

El resultado total del coste imputable es de QUINIENTOS TRECE EUROS CON CUARENTA Y DOS CÉNTIMOS DE EURO.

7.2.6.Resumen de costes

Finalmente, la suma de todos los costes asociados al proyecto queda como sigue en la siguiente tabla:

Concepto	Importe (en €)
Personal	11.208,66
Coste equipos	513,42
Costes indirectos (20%)	2.344,41
Total sin IVA	14.066,49
Total con IVA (21%)	17.020,45

Tabla 51: resumen de costes del proyecto.

Capítulo 8

8. Conclusiones y trabajos futuros

En este capítulo, se presentan las conclusiones a la finalización del proyecto y los posibles trabajos futuros a realizar.

8.1. Conclusiones

En el primer capítulo de este documento se citaban los objetivos de los que constaba el proyecto fin de carrera, en esta sección se realiza un análisis del trabajo realizado, describiendo los objetivos alcanzados y las conclusiones obtenidas al respecto.

El objetivo principal de este proyecto era la elaboración de una plataforma a medida de almacenamiento secundario, adaptado a las necesidades específicas de aulas con terminales GNU/Linux en un entorno universitario. Este objetivo ha sido la base de todo el proyecto, obteniendo como resultado un sistema de almacenamiento con arquitectura NAS fácilmente integrable. Se ha cumplido el objetivo principal, ya que dicho sistema ha sido capaz de proporcionar almacenamiento a varios terminales GNU/Linux. Para la realización del proyecto se ha contado con la colaboración de la Universidad Rey Juan Carlos, tanto para obtener recursos físicos como en acceso a información real de uso de terminales.

Además, también se han alcanzado los objetivos secundarios definidos. El primero de ellos, comprensión de las necesidades y características referentes al almacenamiento secundarios de terminales GNU/Linux, creando un estudio del comportamiento de las aulas actuales de la URJC.

El segundo de ellos, estudio de las tecnologías disponibles en el contexto del software libre para la creación de sistemas de almacenamiento, mediante la lectura y estudio en profundidad de documentación.

El tercero de ellos, adquirir conocimientos técnicos necesarios para el manejo de componentes hardware y software que forman el sistema. La realización de los estudios ha requerido montar el servidor desde cero en el propio centro de datos del grupo de investigación GSyC/Libresoft.

Y por último, el cuarto de ellos, conocer cómo afectan al rendimiento diferentes factores relacionados con el manejo de los datos, mediante la ejecución de la herramienta de *benchmark* IOzone sobre un amplio abanico de posibles configuraciones. La realización de este objetivo también ha permitido identificar factores a tener en cuenta en el diseño del sistema, los cuales, no habían sido identificados durante la realización del segundo objetivo secundario. Un ejemplo son las configuraciones de *page cache*.

Como conclusión general, se han logrado los objetivos propuestos en este proyecto y, adicionalmente, se han creado procedimientos que permiten replicar el sistema diseñado de forma rápida, haciendo fácilmente reproducible el presente proyecto.

Finalmente, durante el proyecto, han sido de particular utilidad los conocimientos adquiridos en asignaturas relacionadas con la especialidad de aplicaciones y sistemas distribuidos. Estas asignaturas son arquitectura de computadores, redes, sistemas operativos y desarrollo de aplicaciones distribuidas. En cambio, ha sido necesario aprender, en profundidad, aspectos técnicos relacionados con los productos utilizados. Esto no ha sido un problema, gracias a la base sólida de conocimientos que ha aportado los estudios cursados.

8.2. Trabajos futuros

El presente proyecto ha logrado un sistema de almacenamiento para terminales a partir del cual se pueden contemplar varias líneas de posibles trabajos futuros. Este proyecto es el núcleo de otros posibles diseños que, con más recursos, se pueden llegar a desarrollar.

Por este motivo, se pueden mejorar diversos aspectos relacionados con la escalabilidad del sistema, mantenimiento, disponibilidad y la confidencialidad de los datos.

El uso de cuotas de usuario es un aspecto importante en este tipo de servicios, implica la identificación de usuarios por parte del servidor de almacenamiento y, por lo tanto, la integración del mismo con el servicio de autenticación, normalmente LDAP, ya existente en las aulas. Trabajar en este sentido mejoraría la escalabilidad del sistema, permitiendo dar servicio a más usuarios e impidiendo que los posibles usos abusivos provoquen problemas que repercutan en la disponibilidad.

El uso de sistemas de ficheros compartidos como OCFS2 puede ayudar a basar la arquitectura de almacenamiento en más de un nodo, esto permitiría repartir la carga

entre varios servidores y, por lo tanto, soportar más aulas de terminales GNU/Linux. Añadir más nodos sería trivial, por lo que el uso de estas tecnologías beneficiaría a la escalabilidad del mismo.

A su vez, desacoplar el manejo de los dispositivos de bloques de los sistemas de ficheros, mediante el uso de una arquitectura SAN basada en el protocolo iSCSI, permitiría una administración más especializada. Gracias a esto, se tendría una arquitectura en la que el manejo de dispositivos de bloques y RAIDs se haría en unos servidores, y el de sistema de ficheros en otros. Esta línea de trabajo mejoraría el mantenimiento y la escalabilidad, sobre todo en entornos de mayor demanda que el planteado en este proyecto.

A pesar de que el sistema desarrollado soporta algunos fallos, como la pérdida de corriente de una fuente y la pérdida de un disco en cada uno de los RAID, cosas peores podrían suceder. Por este motivo, junto con la utilización de DRBD para duplicar dispositivos de bloques, se pueden incluir sistemas de alta disponibilidad como Pacemaker. Este sistema es capaz de detectar falta de servicio de uno o varios recursos, asumiendo automáticamente la tarea de utilizar el nodo duplicado para continuar con el servicio. Esta línea de trabajo mejoraría el sistema propuesto en este proyecto en el aspecto de la disponibilidad.

Finalmente, dado que el sistema es utilizado por muchos usuarios que almacenan información de todo tipo, se podría mejorar el aspecto de la confidencialidad. Para ello, habría que mejorar el sistema de autenticación de clientes que quieren importar, mediante NFS, el sistema de ficheros. El uso *netgroups* de NIS permite definir grupos de terminales de forma inequívoca. De este modo, no basta con pertenecer a la red del servidor para poder importar sistemas de ficheros, sino que, además, el cliente debe estar identificado y definido en un grupo de terminales válidos.

Anexo A - Resultados de la encuesta de integración

En este anexo se indican los emails recibidos por parte de algunos profesionales del sector. En estos emails se contesta a la pregunta de si consideran que el sistema, diseñado en este proyecto, es fácil de integrar en un sistema de aulas existente.

En continuación el email que se les mandó:

from: **Roberto Andradas Izquierdo** <randradas@gmail.com>

date: 30 June 2013 23:22

subject: Cuestión sobre mi PFC

Hola,

estoy realizando mi PFC, es un servidor de almacenamiento por red
para dotar de almacenamiento a aulas GNU/Linux.

Tras diversos análisis de rendimiento he concluido una determinada
solución a medida que me permite exportar por NFS un sistema de ficheros
para albergar los homes de los usuarios.

Para integrar mi sistema como nuevo servidor de unas aulas GNU/Linux
es necesario realizar algún cambio en los clientes. Estos cambios,
en realidad, consisten en una única modificación en el fichero /etc/fstab.

Dichos terminales ya tienen instalado el software
necesario para hacer de clientes NFS.

- ¿consideras que es complicado en exceso realizar esta modificación en
el hipotético caso de que tuvieras que hacerlo tú en los PCs de las
aulas que nombro?

- ¿crees que podría hacerse de otra forma más sencilla?

Un saludo y muchas gracias

A continuación las respuesta de cada uno de ellos.

Alberto Cacharro – Experto IT en Departamento Sistemas Transaccionales NoJava Boadilla de Produban (empresa del grupo Santander).

from: **Alberto Cacharro** <acacharro@gmail.com>

to: Roberto Andradas Izquierdo <randradas@gmail.com>

date: 1 July 2013 08:57

subject: Re: Cuestión sobre mi PFC

Hola Rober,

Los clientes son linux? si son linux obviamente no me parece ningún tipo de problema. Supongo que en tu infraestructura de servidor para los home dir lo tendrás redundado y con algun tipo de raid no (com poco mirro+1 deseable mirror 5).

Juan José Amor Iglesias – Responsable de Unix en ASEFA Seguros y Reaseguros, S.A.

from: **Juan Jose Amor Iglesias** <jjamor@gmail.com>

to: Roberto Andradas Izquierdo <randradas@gmail.com>

date: 1 July 2013 08:17

subject: Re: Cuestión sobre mi PFC.

modificar el /etc/fstab no es un problema gordo, lo único gordo es tenerlo que hacer en cientos de puestos. En la administración de sistemas de muchas máquinas lo que se intenta es mantener estos cambios de forma centralizada, en Windows ya lo logran con la GP (Política de Grupo), que consiste en según el equipo y/o usuario que se conecte, obligar al equipo cliente a adoptar determinadas configuraciones impuestas en el controlador de dominio. De este modo, por ejemplo, se obliga a que el equipo mapee determinadas unidades de red o ponga una determinada configuración en el navegador (configuración de proxy

empresarial y cosas así). En sistemas Unix, no pensados inicialmente para equipos de escritorio, esto nunca se ha logrado de forma estándar. Pero en los últimos tiempos tenemos soluciones tipo chef o puppet que emulan esa característica. De nuevo se trata de obligar al equipo cliente a configurarse de una determinada manera al conectarse a la red, esto incluye ejecución de cualquier tipo de script que sirva por ejemplo para el propósito que dices. Pero vamos, el principal problema de esto es que no existe todavía un verdadero estándar de facto suficientemente desarrollado como para que se adopte con facilidad y mínimo esfuerzo en las organizaciones.

José Antonio Torrado Navas – Administrador de aulas GNU/Linux de la Univerisdad Rey Juan Carlos.

from: **jose antonio torrado navas** <jtorrado@gsyc.es>

to: Roberto Andradas Izquierdo <randradas@gmail.com>

date: 1 July 2013 10:44

subject: Re: Cuestión sobre mi PFC

Hola roberto, te respondo.

En nuestro caso tenemos puppet, esta herramienta nos permite tener un estado homogoneo en los ordenadores. Bastaría con modificar el archivo fstab del sistema puppet y este se replica a todos.

Un saludo.

Anexo B - Procedimientos

1. Procedimiento general de creación de RAID

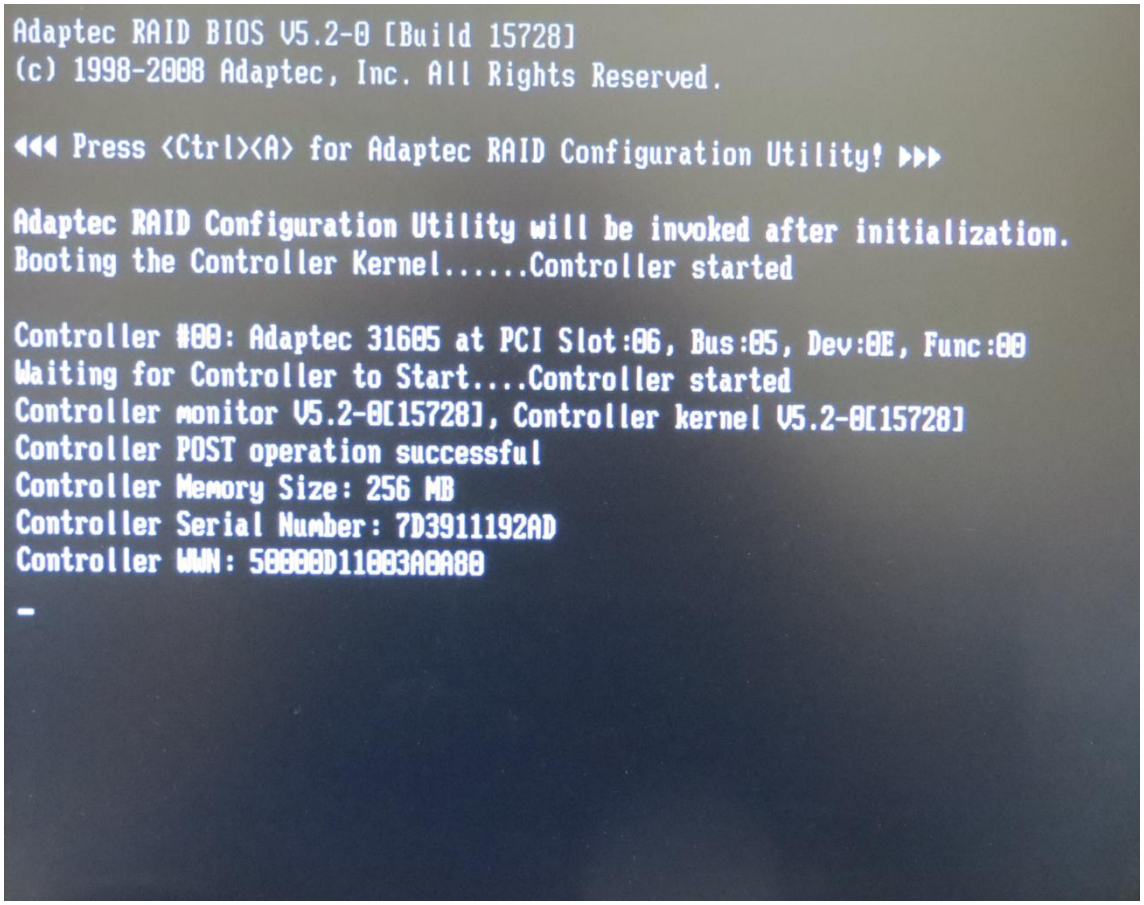

Ilustración 52: creación de RAID (1).

Durante el encendido del servidor se debe pulsar **CTRL+A** para entrar en la aplicación que permite configurar la controladora RAID.

Ilustración 53: creación de RAID (2).

Una vez en la aplicación se selecciona la opción **Array Configuration Utility** que permite realizar la gestión del RAID de discos.

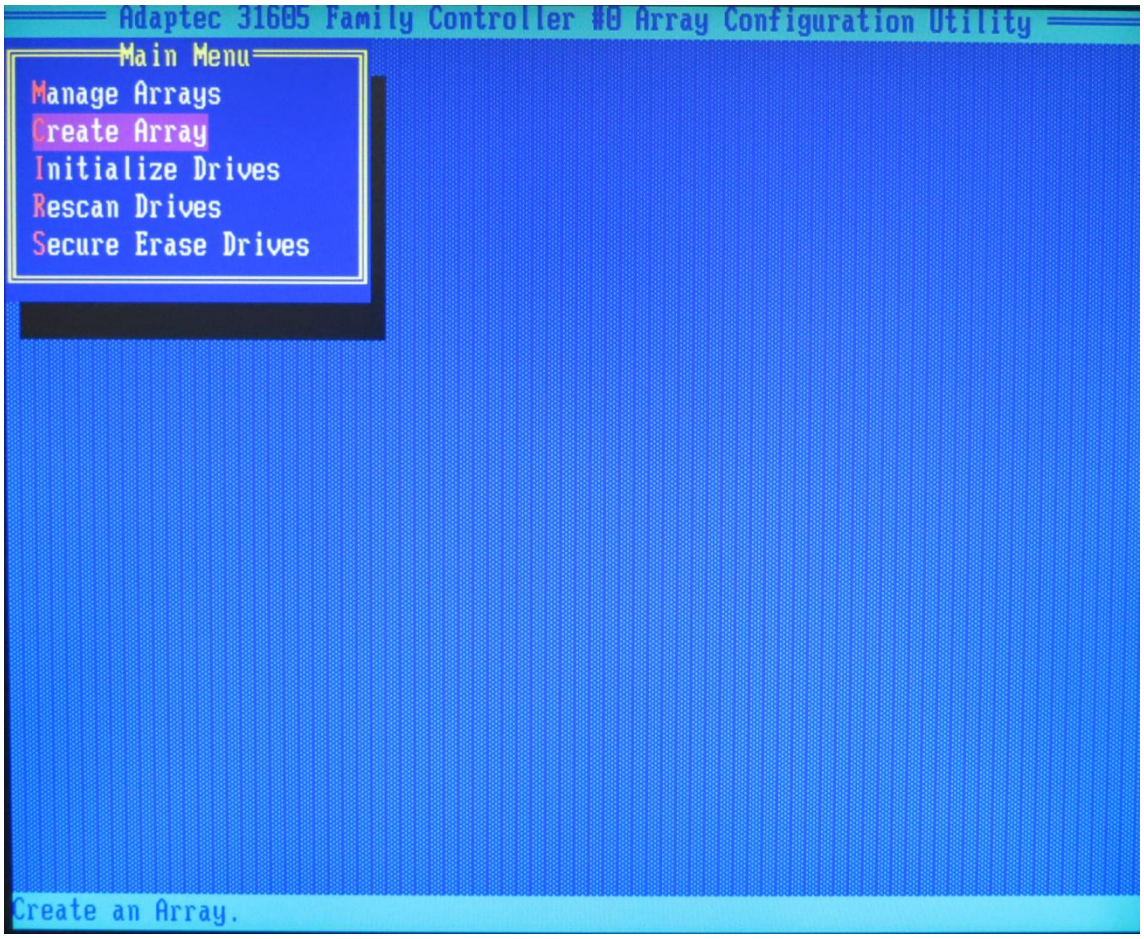

Ilustración 54: creación de RAID (3).

Se procede a la creación del RAID eligiendo la opción **Create Array**.

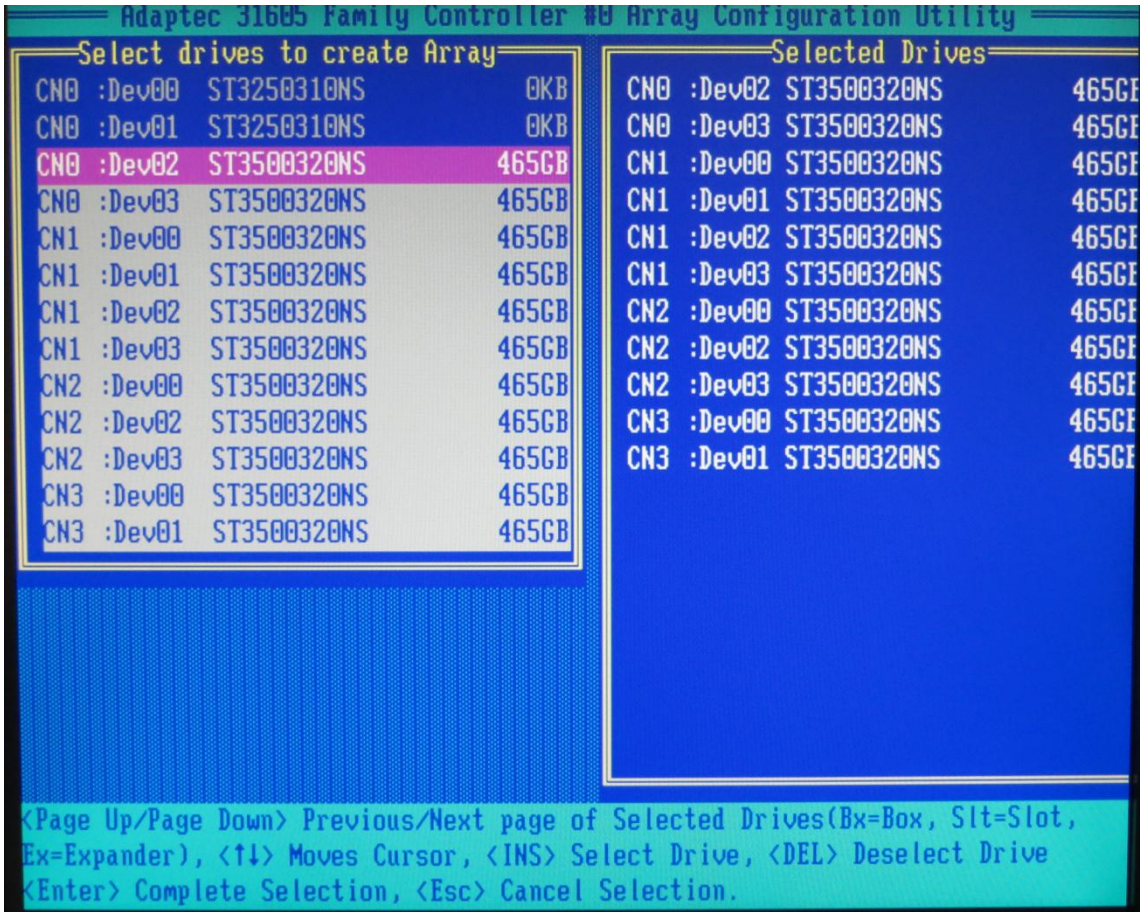

Ilustración 55: creación de RAID (4).

A continuación, se deben seleccionar los discos implicados en el RAID pulsando la tecla INS y posteriormente pulsando **Enter** para finalizar la selección y acceder al siguiente paso.

Como se puede observar en esta imagen, los discos seleccionables son aquellos que no forman parte de un RAID existente. En el momento de capturar esta imagen ya existe un RAID formado por los dos primeros discos ya que son utilizados para albergar el sistema operativo del servidor, el resto de discos son seleccionados para crear el RAID destinado a almacenamiento por red sobre el cual se harán las pruebas.

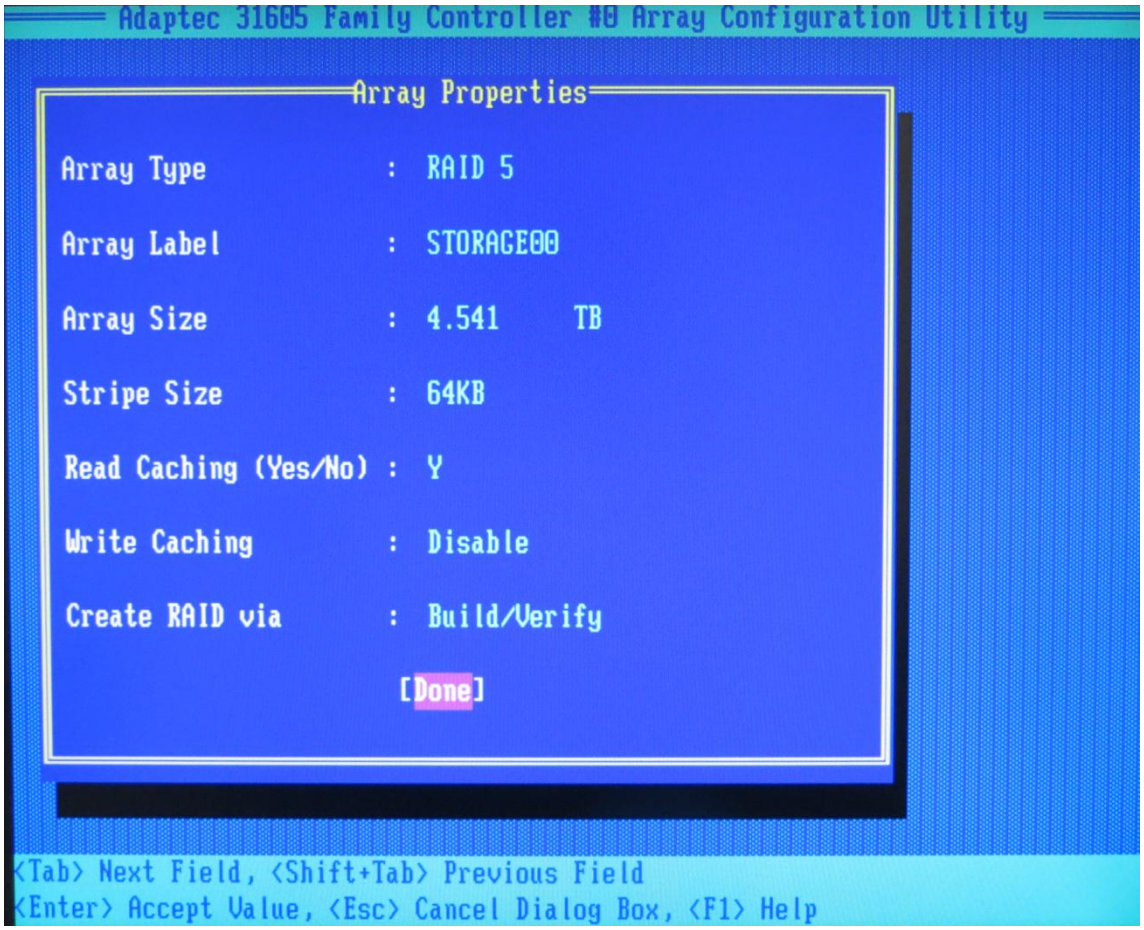

Ilustración 56: creación de RAID (5).

En este paso se deben seleccionar las características del RAID a configurar. Aparecen las siguiente opciones:

- **Array Type**: tipo de RAID, será uno de los indicados en Tabla 25: configuraciones de RAID a prueba.
- **Array Label**: se debe indicar una etiqueta al RAID para diferenciarlo de otros posibles RAID.
- **Array Size**: el sistema calcula el espacio útil disponible para albergar un sistema de fichero en función del número de discos, su capacidad y el tipo de RAID seleccionado en la primera opción.
- **Stripe Size**: es el tamaño en bytes de los fragmentos en los que será dividido un dato, cada fragmento será asignado a un disco del RAID. De este valor

depende el tamaño de stride y stripe-width del sistema de ficheros comentado previamente en esta sección.

- **Read Caching**: la controladora RAID ofrece cache de datos para acelerar la lectura de información que ha sido leída recientemente.

- **Write Caching**: la controladora RAID ofrece también cache para las escrituras permitiendo escrituras más rápidas que posteriormente y a espaldas del sistema de ficheros serán volcadas a disco. En el manual de la controladora [40] se indica que el uso de esta característica puede provocar escrituras incompletas en caso de fallo de energía en el servidor y por lo tanto pérdida de datos.

- **Create RAID via**: la creación del RAID no es inmediata y requiere tiempo en función del número de discos. Esta opción permite especificar diferentes formas de configurar el RAID para, por ejemplo, construirlo mientras se comienza a utilizar, construirlo evitando la aparición del dispositivo RAID hasta que no está terminado y evitar su uso y otras opciones. Todas las opciones son válidas siempre y cuando las pruebas se realicen con el RAID completamente construido ya que de lo contrario su rendimiento no es óptimo y la prueba no es válida.

Una vez seleccionadas las opciones se debe pulsar **Done** para comenzar la creación del RAID.

2. Procedimiento general de creación de sistema de ficheros

```
:~# mkfs.ext4 -b 4096 -E stride=16,stripe-width=167 /dev/sdb1
```

$$stride = \frac{p}{b}$$

$$stripe\ width = \frac{d \times p}{b}$$

p – Stripe Size
d – Número de discos para escrituras
b – Tamaño de bloque del sistema de ficheros

En el caso del RAID 5, el valor de *d* es uno menos que el número total de discos configurados en el RAID ya que durante las escrituras uno de los discos siempre es utilizado para escribir la paridad, ver sección 2.2.1.3.

3. Ejecución de IOzone

3.1. Rendimiento

```
:/data# iozone -Ra -g 16GB > ~/logs/raid5_12_64_4096.stdout
```

A continuación, una descripción de los modificadores aplicados a iozone para su ejecución:

- **–R**: pide que genere una salida compatible con hojas de cálculo, es decir, campos separados por comas.
- **–a**: modo automático. Genera una salida que cubre todas las operaciones testeadas.
- **–g**: se elige el tamaño máximo de fichero sobre el que se harán las pruebas de rendimiento.
- >: redirección de la salida estándar a un fichero de texto que sigue el formato descrito en la sección 4.3.2.1.

Merece la pena comentar el valor aplicado a –g en este ejemplo. Debido al funcionamiento del sistema de entrada y salida del sistema operativo GNU/Linux, ver sección 2.2.2, el sistema tratará de almacenar en memoria las partes de los ficheros sobre los que se están realizando operaciones para agilizar el proceso y consumir menos tiempo desde que se solicita una escritura/lectura hasta que se da por realizada. El servidor de almacenamiento dispone de una cantidad total de 8GB de memoria RAM por lo que es importante saber cómo se comportaría el sistema para ficheros que no pueden ser guardados en memoria y para ello, como recomienda uno de los documentos

proporcionados por IOzone [33], se ha elegido un valor que duplica el de la memoria disponible.

3.2. Latencia

```
:/data# iozone -RQa -g 8GB > ~/logs/iozone_latency.stdout
```

Dado que el resto de modificadores mostrados en este comando son las mismas que las de la sección anterior únicamente se comenta la opción –Q.

- **-Q**: pide a IOzone que realice medidas de latencia. IOzona las realiza únicamente para las operaciones *read*, *write*, *re-read*, *re-write*.

4. Creación de partición /dev/sdb1

A continuación se muestra el procedimiento a seguir para crear la partición sdb1 en el dispositivo /dev/sdb. Basta con introducir la siguiente secuencia de comandos en el servidor.

```
:~# parted /dev/sdb mklabel gpt
:~# parted /dev/sdb print
```

A continuación se muestra información sobre el dispositivo de bloques */dev/sdb*. Es importante fijarse en el tamaño del dispositivo, ya que será utilizado para la creación de la partición.

```
Model: Adaptec STORAGE00 (scsi)
Disk /dev/sdb: 5492GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
```

Se introduce el siguiente comando:

```
:~# parted -a optimal /dev/sdb mkpart primary 0GB 5492GB
```

5. Configuración de page cache

A continuación se muestra el procedimiento a seguir aplicar la configuración indicada a *page cache*. Basta con introducir la siguiente secuencia de comandos en el servidor.

```
:~# emacs /etc/sysctl.conf
```

Añadir las siguientes lineas al final del fichero:

```
vm.dirty_background_ratio = 5
vm.dirty_ratio = 60
```

Cargar la nueva configuración dinámicamente en el kernel:

```
:~# sysctl -p
```

6. Creación de sistema de ficheros en /dev/sdb1

A continuación se muestra el procedimiento a seguir para crear el sistema de ficheros de la partición */dev/sdb1*. Basta con introducir la siguiente secuencia de comandos en el servidor.

```
:~# mkfs.ext4 -b 4096 -E stride=32,stripe-width=352 /dev/sdb1
```

Tras una espera de varios minutos el sistema terminará la tarea, en esete instante se debe configurar el periodo de chequeo automático del sistema de ficheros mediante el siguiente comando:

```
:~# tune2fs –c 26 –i 60d /dev/sdb1
```

Si es necesario que el sistema de ficheros se monte cada vez que el sistema inicia se aplicarán los siguientes comandos:

```
:~# ls –l /dev/disk/by-uuid
2013-06-27 14:41 2759ea8b-9085-400a-93b4-f199fa5b33df -> ../../sda1
2013-06-28 17:48 36810f29-b201-4e97-bce6-0176fcf54ec2 -> ../../sdb1
:~# emacs /etc/fstab
```

Anotando el identificador único del dispositivo sdb1, se introduce la siguiente línea en el fichero que abre el editor:

```
UUID=36810f29-b201-4e97-bce6-0176fcf54ec2 /data/ ext4 \
defaults,noatime,errors=remount-ro 0 2
```

Finalmente, para montar el sistema de ficheros introducido en /etc/fstab basta con crear el punto de montaje, aplicar el comando *mount* y configurar los permisos adecuados de acceso:

```
:~# mkdir /data/
:~# mount –a
:~# chmod 755 /data/
```

7. Exportación de sistema de ficheros mediante NFS

A continuación se muestra el procedimiento a seguir para exportar mediante NFS el sistema de ficheros del dispositivo de bloques de */dev/sdb1*. Basta con introducir la siguiente secuencia de comandos en el servidor.

```
:~# apt-get update && apt-get install nfs-kernel-server
:~# emacs /etc/exports
```

Se introduce la siguiente linea en el fichero que abre el editor:

```
/data/ 193.147.49.0/24(rw,async,no_subtree_check,auth_nlm,mp,root_squash)
```

Se edita el fichero */etc/default/nfs-kernel-server* asignando el siguiente valor a la variable RCPNFSDCOUNT, este cambio indica que se utilizarán 16 hilos para atender las peticiones:

```
RPCNFSDCOUNT=16
```

Finalmente, se aplican los permisos correctos al punto de montaje exportado y se reinician los procesos que gestionan las peticiones NFS.

```
:~# service nfs-kernel-server restart
```

8. Sincronización de datos entre dos servidores

A continuación se muestra el procedimiento para realizar un copia de los datos almacenados en el servidor.

```
:~# cd /data/
```

```
:~# rsync –avz –e 'ssh –p 20022' \
root@193.147.49.11:/disks/raid/homes.fuenla/* .
```

9. Montaje del sistema de ficheros NFS por un cliente GNU/Linux

A continuación se muestra el procedimiento para montar en un cliente GNU/Linux (epsilon02) un sistema de ficheros remoto exportado por NFS.

```
epsilon02:~# emacs /etc/fstab

193.147.49.10:/data/  /home/  nfs  \
rw,rsize=8192,wsize=8192,noatime,hard,intr,tcp,bg 0 2

epsilon02:~# mount -a
```

Glosario de términos

ATA	Advanced Technology Attachment
BSD	Berkeley Software Distribution
BTRFS	B-Tree File System
CAPEX	CAPital EXpenditured
CA	Central Arimética
CC	Central de Control
CRC	Cyclic redundancy check
DAS	Direct Attached Storage
DRBD	Distributed Replicated Block Device
EB	Exabyte
eSATA	External SATA
Ext2	Second Extended filesystem
Ext3	Third Extended filesystem
Ext4	Fourth Extended filesystem
FBA	Fixed Block Architecture
GID	Group Identifier
GNU	GNU is Not Unix
GPT	GUID Partition Table
GSyC	Grupo de Sistemas y Comunicaciones
GUID	Globally unique identifier
IaaS	Infrastructure as a service
IBM	International Business Machines
IDEMA	International Disk Drive Equipment and Material Association

iSCSI	Internet SCSI
LDAP	Lightweight Directory Access Protocol
MBR	Master Boot Record
NAS	Network Attached Storage
NFS	Network File System
NTFS	New Technology File System
OCFS2	Oracle File System 2
OPEX	OPErational EXpenditure
PC	Personal Computer
RAID	Redundant Array of Independent Disks
RPC	Remote Procedure Call
RTT	Round Trip delay Time
SAN	Storage Area Network
SAS	Serial Attached SCSI
SATA	Serial ATA
SCSI	Small Computer System Interface
SSD	Solid State Drive
TCP	Transmission Control Protocol
UC3M	Universidad Carlos III de Madrid
UDP	User Datagram Protocol
UID	User Identifier
URJC	Universidad Rey Juan Carlos
USB	Universal Serial Bus
XDR	External Data Representation
XFS	High-preformance journaling file system

Bibliografía

[1] J. v. Neumann, «First Draft of a Report on the EDVAC,» University of Pennsylvania, 1945.

[2] R. M. Stallman, «Sección uno. El proyecto GNU y el software libre,» de *Software libre para una sociedad libre*, traficantes de sueños, 2007, pp. 59-62.

[3] GSyC, «Departamento de Sistemas Telemáticos y Computación (GSyC),» URJC, [En línea]. Available: http://gsyc.es/.

[4] URJC, «Universidad Rey Juan Carlos,» [En línea]. Available: http://www.urjc.es.

[5] GSyC/LibreSoft, «GSyC/LibreSoft,» GSyC/LibreSoft, [En línea]. Available: http://libresoft.es.

[6] J. C. Pérez, F. G. Carballerira, P. d. M. Anasagasti y F. P. Costoya, «Entrada/Salida,» de *Sistemas Operativos, una visión aplicada.*, McGrawHill, 2001, pp. 352-353.

[7] D. D. Levinthal, «Performance Analysis Guide for Intel Core i7 Processor and Intel Xeon 5500 processors,» Intel Corporation, 2008-2009.

[8] gskill, «[Ripjaws] F3-10666CL7T-12GBRH (4Gx3),» gskill.com, [En línea]. Available: http://www.gskill.com/products.php?index=330. [Último acceso: 31 01 2013].

[9] W. Digital, «WD Blue 80 GB PATA Hard Drives (WD800AAJB) Spec Sheet,» [En línea]. Available: http://www.wdc.com/wdproducts/library/SpecSheet/ENG/2879-771436.pdf. [Último acceso: 31 1 2013].

[10] W. Digital, «WD VelociRaptor 1 TB SATA Hard Drives (WD1000DHTZ),» [En línea]. Available: http://www.wdc.com/wdproducts/library/SpecSheet/ENG/2879-701284.pdf. [Último acceso: 31 1 2013].

[11] Heron2/MistWiz, «Disk-structure2,» Wikimedia Commons, [En línea]. Available: http://commons.wikimedia.org/wiki/File%3ADisk-structure2.svg. [Último acceso: 04 02 2013].

[12] J. C. Pérez, F. G. Carballerira, P. d. M. Anasagasti y F. P. Costoya, «Entrada/Salida,» de *Sistemas Operativos, una visión aplicada.*, McGrawHill, 2001, pp. 373-379.

[13] IBM, «IBM System/360 Operating System: System Control Blocks,» IBM, 1973.

[14] IBM, «IBM System/360 Model 20 Functional Characteristics,» IBM, 1967.

[15] IBM, «IBM System/360 Model 44 Functional Characteristics».

[16] P. P. Chicoine, H. G. M. Hassner, C. E. Grochowsky, M. S. Jenness, S. M. Noblitt, B. G. Silvus, W. D. C. Stevens y L. C. B. Weber, «Hard Disk Drive Long Data Sector White Paper,» IDEMA, 2007.

[17] IDEMA, «IDEMA The International Disk Drive Equipment and Materials Association,» [En línea]. Available: http://www.idema.org. [Último acceso: 04 02 2013].

[18] Microsoft, «Basic and Dynamic Disks,» [En línea]. Available: http://msdn.microsoft.com/en-us/library/aa363785%28VS.85%29.aspx. [Último acceso: 04 02 2013].

[19] C. M. Burnett, «File:RAID 0.svg,» [En línea]. Available: http://en.wikipedia.org/wiki/File:RAID_0.svg. [Último acceso: 19 02 2013].

[20] C. M. Burnett, «File:RAID 1.svg,» [En línea]. Available: http://en.wikipedia.org/wiki/File:RAID_1.svg. [Último acceso: 19 02 2013].

[21] C. M. Burnett, «File:RAID 5.svg,» [En línea]. Available: http://en.wikipedia.org/wiki/File:RAID_5.svg. [Último acceso: 19 02 2013].

[22] C. M. Burnett, «File:RAID 6.svg,» [En línea]. Available: http://en.wikipedia.org/wiki/File:RAID_6.svg. [Último acceso: 19 02 2013].

[23] Paulish, «File:RAID 10 6Drives.svg,» [En línea]. Available: http://en.wikipedia.org/wiki/File:RAID_10_6Drives.svg. [Último acceso: 19 02 2013].

[24] R. Love, «The Page Cache and Page Writeback,» de *Linux Kernel Development*, Addison-Wesley Professional, 2010, p. 440.

[25] S. D. Pate, UNIX Filesystems Evolution, Design and Implementation., WILEY, 2003.

[26] J. Corbet, A. Rubini y G. Kroah-Hartman, Linux Device Drivers, O'Reilly, 2005.

[27] Microsoft, «Description of the FAT32 File System,» Microsoft, [En línea]. Available: http://support.microsoft.com/kb/154997. [Último acceso: 22 02 2013].

[28] X. Community, «XFS Papers and Documentation,» XFS Community, [En línea]. Available: http://xfs.org/index.php/XFS_Papers_and_Documentation. [Último acceso: 24 02 2013].

[29] C. Hellwig, «XFS: the big storage file system for Linux,» XFS, 2009.

[30] D. Chinner, «XFS: Adventures in Metadata Scalability,» redhat; linux.conf.au, 2012.

[31] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, S. Microsystems, C. Beame, H. Ltd, M. Eisler, D. Noveck y N. Appliance, «rfc3530,» The Internet Engineering Task Force, 2003. [En línea]. Available: http://www.ietf.org/rfc/rfc3530.txt. [Último acceso: 25 02 2013].

[32] B. Pawlowski, S. Shepler, C. Beame, B. Callaghan, M. Eisler, D. Noveck, D. Robinson y R. Thurlow, «The NFS Version 4 Protocol,» de *Sane*, Maastricht, 2000.

[33] W. D. Norcott, Iozone Filesystem Benchmark.

[34] R. v. Riel, «kernel.org,» [En línea]. Available: https://www.kernel.org/doc/Documentation/sysctl/vm.txt. [Último acceso: 22 06 2013].

[35] T. Ts'o, «mke2fs - create an ext2/ext3/ext4 filesystem».

[36] GossamerThreads, «Linux Kernel Mailing List Archive,» [En línea]. Available: http://www.gossamer-threads.com/lists/linux/kernel/1017204. [Último acceso: 01 07 2013].

[37] C. Ltd, «Ubuntu Licensing,» Canonical, [En línea]. Available: http://www.ubuntu.com/about/about-ubuntu/licensing. [Último acceso: 30 06 2013].

[38] UC3M, «Proyecto fin de carrera, presupuesto.,» [En línea]. Available: http://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20(3)_1.xlsx. [Último acceso: 02 07 2013].

[39] UC3M, "Universidad Carlos III de Madrid," [Online]. Available: http://www.uc3m.es.

[40] Adaptec, «www.adaptec.com,» [En línea]. Available: http://download.adaptec.com/pdfs/user_guides/adaptec_raid_controller_iug_08_2008.pdf. [Último acceso: 31 05 2013].

[41] J. Corbet, A. Rubini y G. Kroah-Hartman, «Classes of Device and Modules,» de *Where the kernel meets the hardware; Linux Device Drivers*, 2005, pp. 5-8.