TecnoCampus
Escola Superior
Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grau en Enginyeria Informàtica de Gestió i Sistemes d'Informació

Implementación De Una Solución De Diskless Workstations Para Empresas

Memoria

JAN LOPERA CANO

TUTOR: EUGENIO FERNÁNDEZ

2021 - 2022

TecnoCampus
Mataró-Maresme

Dedicado a todas aquellas personas que están pasando por un momento complicado, porque nunca hay que rendirse. DEP Liz, se te echa de menos.

Agradecimientos

A mi familia y a mis amigos por el apoyo que me han brindado durante la realización de este proyecto, sin ellos hubiera sido imposible.

A mi tutor, Eugenio Fernández, quien siempre ha confiado en mí y ha apostado por mostrar mis conocimientos al mundo.

Abstract

This project focuses on the development of a comprehensive solution to increase energy efficiency, decrease maintenance costs, and increase the reliability of computers in a company. Throughout the report, the requirements of the solution are explored, a prototype is designed, and the solution is developed and implemented in a scalable working prototype.

Resum

Aquest projecte se centra en el desenvolupament d'una solució integral que permeti a augmentar l'eficiència energètica, disminuir la despesa en manteniment, i augmentar la fiabilitat dels ordinadors en una empresa. Al llarg de la memòria s'exploren els requisits de la solució, es dissenya un prototip, i es desenvolupa la solució implementant-la en un prototip funcional escalable.

Resumen

Este proyecto se centra en el desarrollo de una solución integral que permita a aumentar la eficiencia energética, disminuir el gasto en mantenimiento, y aumentar la fiabilidad de los ordenadores en una empresa. A lo largo de la memoria se exploran los requisitos de la solución, se diseña un prototipo, y se desarrolla la solución implementándola en un prototipo funcional escalable.

Índice

Tablas de Ilustraciones y Figuras

Capítulo I: Introducción y objeto del proyecto

Zygmunt Bauman describe el momento actual de la civilización humana con un concepto llamado *Modernidad Tardía* o *Modernidad Líquida* [1]: un momento en que los conceptos más esenciales para todo ser humano del primer mundo pasan a un estado de liquidez e incertidumbre solo vistos hace miles de años. Sin embargo, la sociedad, lejos de alejarse de la idea y despreciar esta liquidez, parece acogerla, arroparla, y hacerla suya.

Esta falta de estabilidad y solidez que produce este mundo líquido se refleja en cada uno de los aspectos de la vida de las personas y ningún sector de la humanidad está exento de estas consecuencias: la falta de una estructura firme acaba causando un gran incremento de los trabajos temporales [2]. La inestabilidad política y las tensiones entre estados soberanos producen virulentas alteraciones del precio de la energía [3]. El propio aumento del individualismo ha aumentado la ferocidad propia del sistema económico actual y de las grandes corporaciones, que buscan perseguir el mayor beneficio sin importar los medios utilizados.

Todos estos factores han agravado tanto la situación climática como el desenfrenado estilo de vida que se lleva tanto en todos los países del primer mundo: el consumismo está implantado en esta sociedad y no pretende desprenderse de ella tan fácilmente.

Estos valores líquidos también han penetrado con fuerza en el mundo de la informática y los videojuegos: los clásicos valores que perseguían optimizar al máximo los sistemas y mejorar el rendimiento energético y económico parecen haberse conservado solo en la mente de los ingenieros que supervisan los grandes centros de datos del planeta.

La poca preparación de la sociedad ante eventos catastróficos, como una pandemia, han dejado en evidencia que no solo pocas empresas están preparadas para el teletrabajo, si no que las pocas que lo están sufren de multitud de problemas derivados de la falta de

infraestructura, la falta de recursos, una mala eficiencia de los sistemas, y una falta evidente de técnicos de primer y segundo nivel que ayuden a los empleados.

El objeto principal de este proyecto es diseñar e implementar una solución informática que permita aumentar la eficiencia, tanto energética como a nivel de mantenimiento y de facilidad de acceso, y que sea adaptable a la flexibilidad laboral y a la temporalidad de los empleos.

Se pretende conseguir este objeto mediante el uso de dispositivos Raspberry Pi, que se pretenden usar como terminales; uno (o varios) servidores de virtualización, que se encargarán de alojar las estaciones de trabajo de los empleados de las empresas; y el uso de proyectos de código abierto para el encaminamiento IP. También se pretende diseñar la arquitectura de red que requeriría una hipotética empresa media para poder implementar la solución, además de la puesta en marcha de los diferentes servicios necesarios para el correcto funcionamiento del sistema.

Capítulo II: Objetivos

El objetivo principal de este proyecto es el desarrollo de un sistema que permita, a una empresa mediana, sustituir los ordenadores de sus trabajadores por clientes ligeros que proporcionen ahorro energético, ahorro en mantenimiento, un menor tiempo de downtime, y capacidad para teletrabajar.

2.1 Objetivos del producto

Se han detallado los siguientes objetivos primarios del producto:

1. Desempaquetar una Raspberry Pi acabada de comprar, conectarle teclado, ratón, monitor, y cable de red, y que automáticamente (sin necesidad de tarjeta microSD) se tenga disponible una estación de trabajo.
2. Conectarse de forma automática a una máquina virtual sin necesidad de conocer su nombre de dominio o su IP.
3. Conectarse al puesto de trabajo de la empresa desde una red externa de forma segura.

Se han detallado el siguiente objetivo secundario del producto:

1. Asegurar un mayor índice de seguridad en la infraestructura.

Estos objetivos se han agrupado en tres metas diferentes:

- Arrancar un dispositivo Raspberry Pi por red
    - Desempaquetar una Raspberry Pi acabada de comprar, conectarle teclado, ratón, monitor, y cable de red, y que automáticamente (sin necesidad de tarjeta microSD) se tenga disponible una estación de trabajo.
    - Asegurar un mayor índice de seguridad en la infraestructura.

- Desarrollar el conector (Distribución Linux Personalizada)

  - Desempaquetar una Raspberry Pi acabada de comprar, conectarle teclado, ratón, monitor, y cable de red, y que automáticamente (sin necesidad de tarjeta microSD) se tenga disponible una estación de trabajo.
  - Asegurar un mayor índice de seguridad en la infraestructura.

- Creación de un servicio “DHCP” para VMs
  - Desempaquetar una Raspberry Pi acabada de comprar, conectarle teclado, ratón, monitor, y cable de red, y que automáticamente (sin necesidad de tarjeta microSD) se tenga disponible una estación de trabajo.
  - Conectarse de forma automática a una máquina virtual sin necesidad de conocer su nombre de dominio o su IP.
  - Conectarse al puesto de trabajo de la empresa desde una red externa de forma segura.

2.2 Cliente potencial

Hay dos perfiles de cliente potencial del producto que pretende generar este proyecto.

El primer perfil es el del director o administrador de una empresa pequeña o mediana que busca delegar la gestión de la infraestructura en una empresa externa, quiere reducir en el mantenimiento de los equipos, o simplemente es un entendido de la materia a nivel no-profesional y busca que su empresa esté actualizada.

El segundo perfil es el de un departamento de IT de una mediana o gran empresa que busque reducir costes en mantenimiento y energía, asegurar mayor disponibilidad de su instalación, resolver incidencias de forma rápida, y habilitar el teletrabajo de forma sencilla.

2.3 Objetivos del cliente

De los clientes potenciales descritos anteriormente se puede elaborar una lista de objetivos que el cliente potencial quiere ver cumplidos gracias a la solución que propone este proyecto.

1. Delegar gestión de la infraestructura
2. Reducir el gasto en mantenimiento
3. Reducir las horas de mantenimiento
4. Reducir el downtime de su infraestructura
5. Reducir el consumo energético
6. Mantener sus equipos actualizados
7. Resolver incidencias de forma rápida y precisa
8. Habilitar el teletrabajo

2.4 Alcance del proyecto

Este proyecto pretende ser una prueba de concepto del sistema propuesto. En ningún momento se garantiza que llegue a un estado de estar listo para ser implementado en un escenario de producción real en el tiempo establecido. Cuando la prueba de concepto sea funcional, momento en el que se considerará listo para ser implementado en un entorno de desarrollo, se darán por cumplidos los objetivos antes mencionados.

Este sistema está diseñado para que los pocos factores limitantes de su escalabilidad sean elementos ajenos al propio diseño. Si se desea escalar el sistema, es posible que se requiera una actualización de la infraestructura de red de la empresa para garantizar una experiencia óptima en todos los escenarios. Esta actualización no se encuentra contemplada en el alcance del proyecto; sin embargo, sí se proveerá, mediante la memoria final, de unos valores de referencia necesarios por cada usuario que quiera ser migrado a la solución descrita en este proyecto.

Capítulo III: Marco Teórico

El mundo de la informática siempre ha estado en constante y rápida evolución. Desde la aparición de los primeros ordenadores de válvulas de vacío, en 1946, hasta la comercialización del primer ordenador con un microprocesador, en 1974 de la mano de Sord [4], ha pasado poco menos de 30 años. Ahora, casi 50 años después, podemos encontrar que el dispositivo con el microprocesador más barato del mercado funciona aproximadamente veinte veces más rápido [5], posee más instrucciones, más memoria, y es infinitamente más barato.

Tom Forester formuló una comparativa similar, pero con el mundo del automóvil: <<Si la automoción hubiera experimentado un desarrollo parecido a la informática, se podría disponer de un Rolls-Royce por menos de 300 pesetas y, además, el vehículo dispondría de la potencia de un transatlántico como el Queen Elizabeth para ser capaz de recorrer un millón de kilómetros (unas 25 vueltas al mundo) con sólo un litro de gasolina>> [6].

Esta evolución, junto con el aumento de la producción y el abaratamiento de los costes, ha causado que nos encontremos con tres etapas con filosofías diferentes en la informática [7]:

- Una primera etapa, donde se tenía un gran ordenador que era compartido con diversas personas mediante el uso de terminales o clientes ligeros.
- Una segunda etapa, donde las personas comenzaban a poseer un ordenador personal y se establecía una relación 1:1 con la máquina.
- Una tercera etapa, donde las personas poseen varios ordenadores y esta es su mayor fuente de interacción: hay ordenadores integrados en los coches, en los relojes, en los teléfonos…

Esta tercera etapa, en la que la humanidad se encuentra actualmente, trae consigo ciertas desventajas que son inherentes a su definición. Este TFG trata de mirar al pasado y rescatar la visión y la filosofía de la primera etapa e incorporarla en los sistemas de la tercera, con el fin de ofrecer lo mejor de ambas.

3.1 Contexto y antecedentes

Aristóteles bautizó al ser humano como un ser social [8], un ser que busca crear lazos y comunidades. Con la aparición de las primeras comunidades sedentarias, llegó la necesidad de comunicación entre las diferentes aldeas que se formaron. Estas aldeas pronto aprenderían a convivir, formando así pequeñas naciones que buscaban expandirse.

El Imperio Romano, uno de los mayores imperios de la historia, tuvo muy presente que la comunicación entre Roma y sus provincias era esencial. El emperador Augusto decidió crear un sistema de transmisión de mensajes, llamado *Cursus Publicus* [9], que se encargaría de transmitir los mensajes entre las diferentes provincias de forma rápida y segura. Al principio, el sistema imitaba una red similar operada por el imperio persa, pero pronto se sustituyó por un sistema de diseño propio que usaba como fuerza fundamental una serie de mozos y caballos. Se aproxima que los mensajes podían llegar a viajar a una velocidad de cien kilómetros por día como máximo.

Con la llegada de la modernidad, las necesidades de comunicación en las personas siguieron aumentando. De esa necesidad de comunicación surgió el telégrafo [10].

Lentamente, los primeros telégrafos ópticos fueron sustituidos por los eléctricos, y estos fueron modernizados y, en cierto modo, automatizados mediante el uso de teletipos.

Del Teletipo al Terminal Gráfico

El teletipo fue todo un éxito en el mundo de la telegrafía, ya que presentaba una forma mucho más sencilla y cómoda de interacción entre el emisor y el receptor. Debido a la popularidad de este dispositivo, y a su facilidad de uso; se decidió darle un uso nuevo hasta la época: hacer de interfaz entre un humano y un ordenador.

Al principio, la interacción humano-ordenador residía en el uso de tarjetas perforadas, cintas magnéticas, o diversos pulsadores que conectabas y desconectaban circuitería del

dispositivo. Una interacción humano-máquina similar a la que se tiene hoy en día parecía impensable; sin embargo, el teletipo acercó ligeramente esa interacción.

Con el teletipo, el usuario era capaz de escribir comandos y esperar una respuesta relativamente inmediata por parte de la máquina.

Pronto, los teletipos mecánicos, similares a una máquina de escribir, fueron sustituidos por teletipos gráficos; que incluían una pantalla de tubo de rayos catódicos como visor en vez del papel.

Con la aparición de los microordenadores, los antiguos teletipos gráficos fueron sustituidos por los propios ordenadores, por lo que ya no era necesario un dispositivo distinto para interactuar con el ordenador. A esta revolución, también se le sumó la aparición de las primeras interfaces gráficas de usuario, cuyo concepto sigue presente desde ese momento.

3.2 Tecnologías

Cuando en las secciones anteriores se habla de mezclar conceptos de esa primera etapa de la informática en la actual, en ningún momento se pretende volver a recurrir a máquinas (hoy en día relativamente obsoletas) como el teletipo. La intención es rescatar ese concepto de terminal como dispositivo independiente que se encuentra entre el ordenador y la persona

Para esto, se requiere el uso de diversas tecnologías ampliamente conocidas y eficaces. Estas tecnologías están divididas en diversas secciones en función de su funcionalidad.

3.2.1 Conexión Remota

La conexión remota a un dispositivo no es algo nuevo. Existen diversos protocolos, creados por diferentes empresas, entre los que se analizarán tres [11].

3.2.1.1 RDP

Remote Desktop Protocol (RDP) es un protocolo propietario desarrollado por Microsoft para su sistema operativo Microsoft Windows. A pesar de que este protocolo es propietario y las especificaciones totales son secretas, al estar basado en ITU T.120 [12] existen diversas implementaciones no-oficiales que permiten usar este sistema con dispositivos Linux, entre otros.

3.2.1.2 VNC

Virtual Network Computing (VNC) es un software que utiliza el protocolo RFB (remote framebuffer) para transmitir audio, video, y acciones del usuario entre el servidor y el cliente. Tanto este protocolo como la suite de software son de código abierto y están protegidos bajo la licencia GPL. Inicialmente fue desarrollado por un pequeño laboratorio que fue comprado por AT&T. Esta solución es multiplataforma tanto en su versión de cliente como en su versión de servidor.

3.2.1.3 NX

NX es un software, en un principio propietario, desarrollado por NoMachine [13] que utiliza una versión modificada del protocolo Differential X Protocol Compressor (DXPC).

3.2.1.4 Pruebas

En la USENIX 2002 [11] se mostró al público una serie de pruebas realizadas a diferentes protocolos de conexión a escritorio remoto. Se analizó tanto la capacidad de respuesta en diferentes condiciones como el uso de recursos tanto de la máquina cliente, como del servidor, y también se tomaron métricas del uso de la red.

Para la navegación web, estos fueron los resultados.

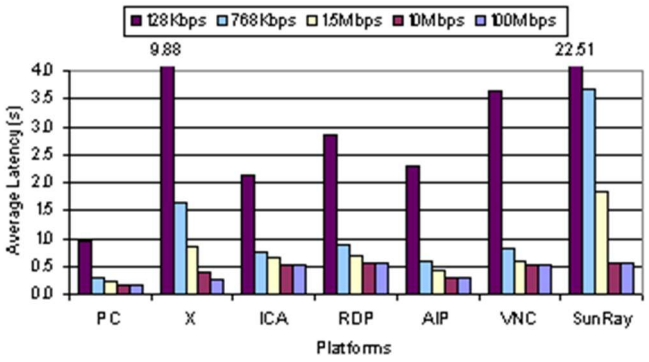

Ilustración 1 Latencia media en la carga de una página web. Fuente: [11]

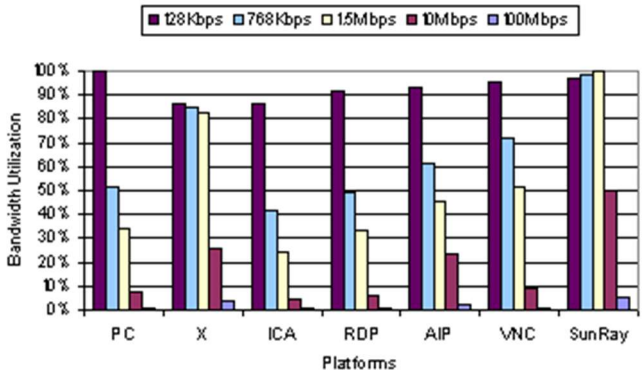

Ilustración 2 Utilización de la red media en la carga de una página web. Fuente: [11]

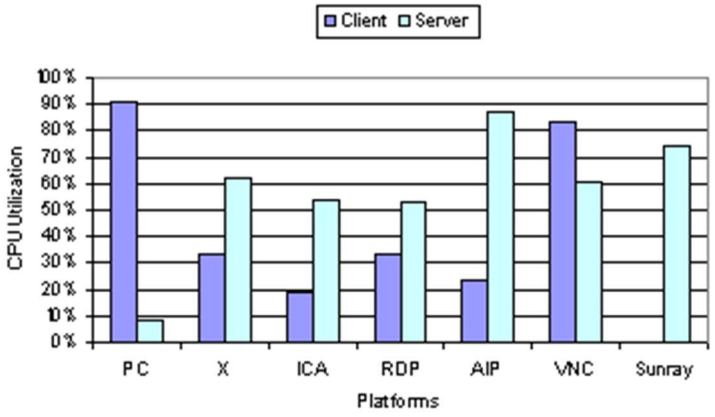

Ilustración 3 Utilización del cliente y servidor media en la carga de una página web. Fuente: [11]

En cuanto a la reproducción de video, los resultados fueron los siguientes.

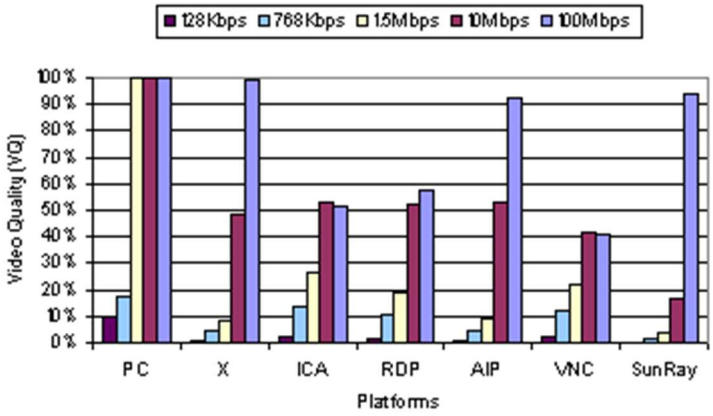

Ilustración 4 Calidad de video media. Fuente: [11]

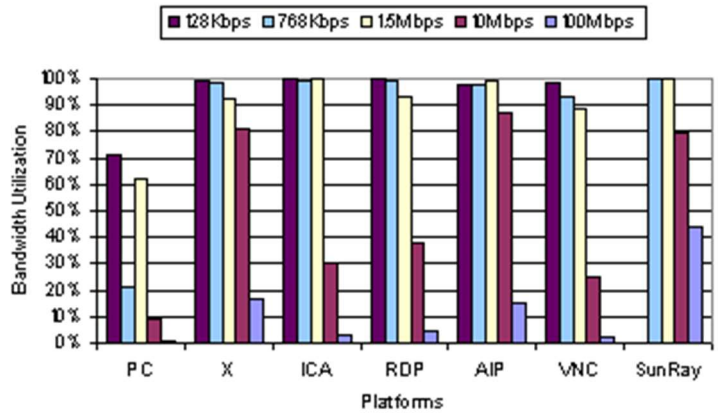

Ilustración 5 Utilización media de la red en la reproducción de video. Fuente: [11]

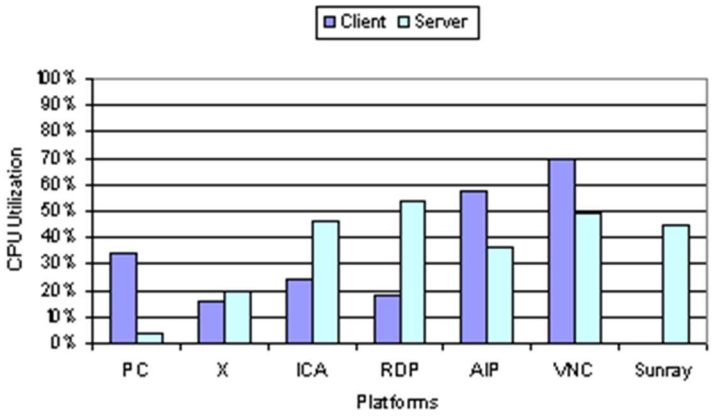

Ilustración 6 Utilización media del cliente y el servidor durante la reproducción de video. Fuente: [11]

3.2.1.5 Conclusiones

Estas pruebas muestran que, teniendo en cuenta el uso que las empresas pretenden dar a este sistema, el mejor protocolo para conexión a escritorio remoto es RDP (si la máquina host es Windows). En segundo lugar, quedaría el software NX, y en tercer lugar VNC.

3.2.2 Virtualización

La virtualización nació como una respuesta a la infrautilización de los recursos de los costosos servidores de la década de los años 60 [14]. Jim Rymarczyk, programador de IBM, decidió intentar virtualizar los servidores de IBM. Esto le llevó a desarrollar el IBM CP-40, que posteriormente evolucionó al CP-67, que permitía el uso de múltiples aplicaciones simultáneamente.

Con la llegada de Unix, y los sistemas operativos preemptivos [15], la utilización de múltiples aplicaciones de forma aparentemente simultanea comenzó a ser una característica común en todos los ordenadores. Fue en ese momento cuando VMware desarrolló su primer hipervisor para permitir que una misma máquina pudiera ejecutar dos sistemas operativos de forma aparentemente simultánea.

La virtualización no es más que la adición de una capa de abstracción entre el hardware de un dispositivo y el sistema operativo huésped. Esta capa de abstracción muestra al sistema operativo una serie de dispositivos virtuales que le permiten operar como si se tratase de una operación normal sobre hardware real. Esta capa de abstracción, llamada Hipervisor, permite que varios huéspedes compartan recursos y operen de forma simultánea sin que los huéspedes sepan con quien comparten recursos o, siquiera, que los están compartiendo.

Esto permite una mayor utilización de los recursos de la máquina debido a que, por norma general, pocas veces se usa durante todo el tiempo de vida de un ordenador el cien por cien de su capacidad de procesamiento.

Para permitir la virtualización en dispositivos x86, se introdujo las extensiones AMD-V e Intel VT-X.

Los dos mejores hipervisores de código abierto actualmente son KVM y Xen [16]. A continuación, se muestra una serie de pruebas entre ambos que pretenden mostrar qué hipervisor es el más indicado para este proyecto. Estas pruebas han sido tomadas de los siguientes trabajos previos: "A quantitative comparison between xen and kvm" [17] y "A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project" [18].

3.2.2.1 Comparativa

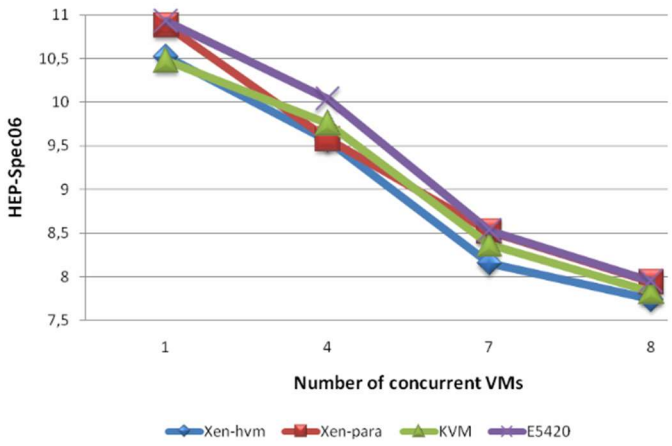

Ilustración 7 Comparativa KVM - Xen en rendimiento según el número de máquinas virtuales. Fuente: [17]

Virtualization Technology	% performance loss (E5420, 8vm)
KVM	3,42
Xen - HVM	4,55
Xen - Paravirtualized	2,02

Tabla 1 Pérdida de rendimiento (8vm) respecto al rendimiento total. Fuente: [17]

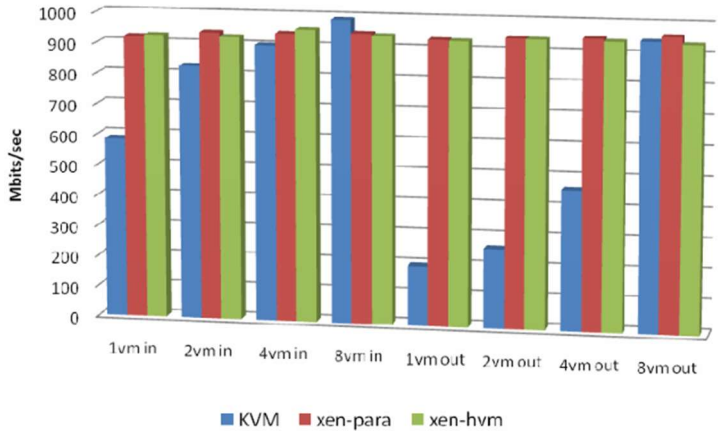

Ilustración 8 Mbits/s máximos capaces de ser transmitidos por el sistema. Fuente: [17]

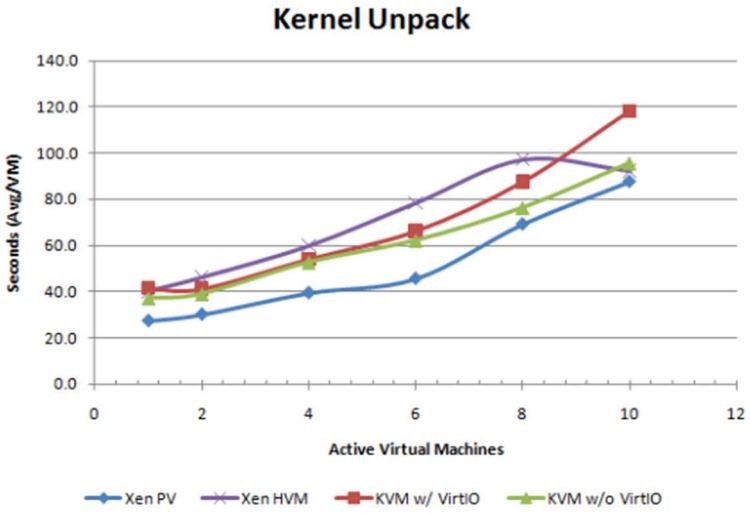

Ilustración 9 Tiempo medio de descompresión de un archivo comprimido según número de VM. Fuente: [18]

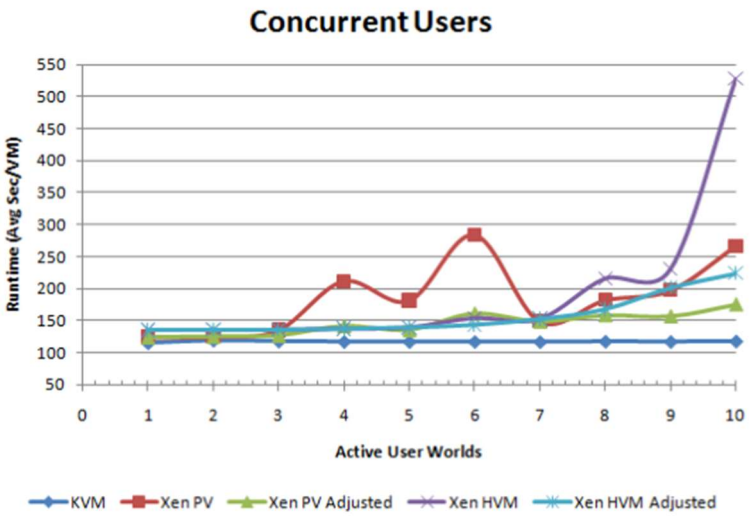

Ilustración 10 Degradación de rendimiento en función de los usuarios concurrentes. Fuente: 18

3.2.2.2 Conclusiones

Ambas alternativas son muy competitivas y ofrecen mejor rendimiento en algunas áreas que la otra; sin embargo, KVM parece tener mejor comportamiento en las tareas que acostumbra a hacer un usuario básico.

3.2.3 Arranque desde red

IBM fue una de las primeras empresas en dar soporte al arranque desde red, con su protocolo Remote Initial Program Load (RIPL), que funcionaba gracias a los servicios aportados por Novel NetWare Control Protocol. RIPL usaba, por ende, redes IPX/SPX.

Las redes Novell IPX/SPX [19] fueron el principal competidor de TCP/IP en los años 90 debido a la popularidad de su producto Novell NetWare. NetWare era un sistema operativo en red, independiente del hardware, que poseía gran cantidad de funcionalidades. Entre las destacadas residen el acceso a ficheros en red, acceso a impresoras en red, sincronización de relojes, y ejecución remota de comandos en servidores. Además, NetWare operaba con una mentalidad diferente a la usada con TCP/IP. TCP/IP asume que la red va a fallar y no confía en su capacidad para transmitir los mensajes; sin embargo, NetWare se basa en la idea de que la red, en condiciones normales, funciona perfectamente [20]. Esto proporciona a NetWare mayor rendimiento que el resto de los sistemas, ya que puede obviar ciertos mecanismos de seguridad y simplificar sus métodos de conexión y envío de paquetes.

Con la caída y paulatina migración de las redes IPX a las redes IP, el protocolo (RIPL) fue sustituido por otro estándar, esta vez creado por Intel, llamado Preboot Execution Environment (PXE) [21]. Su objetivo, a nivel conceptual, es similar al de RIPL, pero ejecutado sobre IP.

Para que PXE funcione correctamente se requiere del protocolo TFTP y de DHCP.

3.2.3.1 DHCP

El protocolo DHCP (Dynamic Host Control Protocol) es el protocolo sucesor de BootP [22]. Este protocolo se encarga de asignar de forma automática, entre otros parámetros, direcciones IP a un host al iniciarse. También es capaz de anunciar la localización de un servidor TFTP desde el que PXE puede descargar una imagen y arrancar.

3.2.3.2 TFTP

El protocolo TFTP (Trivial File Transfer Protocol) es una versión simple y reducida del popular protocolo de transferencia de archivos FTP. Al ser una versión simplificada, es fácil ser embebido en el firmware de PXE para permitir una conexión a un servidor que soporte el protocolo y descargar una imagen de un sistema operativo que, tras ser almacenado en RAM y verificado, se ejecutará en el dispositivo.

3.2.3.3 NFS

NFS es un protocolo, desarrollado por Sun Microsystems, que permite al usuario almacenar ficheros en red [23]. Este protocolo de sistema de ficheros en red está pensado para permitir a los usuarios acceder a los datos como si estuvieran en una unidad local, permitiendo así que el sistema operativo utilice sus comandos clásicos para el manejo de ficheros.

3.2.4 Raspberry Pi

Los dispositivos Raspberry Pi son un conjunto de ordenadores de prestaciones y tamaño reducidos basados en la arquitectura ARM. Estos dispositivos son completamente programables y aceptan numerosa cantidad de software. Entre el software más destacado, se encuentra Raspberry Pi OS.

3.2.4.1 Raspberry Pi OS

Raspberry Pi OS es una distribución Debian modificada para funcionar a la perfección con los dispositivos Raspberry Pi. Este software está protegido bajo una licencia GPL. Al ser una distribución Linux compilada para ARM, se puede usar el famoso compilador GCC para desarrollar aplicaciones, convirtiendo este dispositivo en una gran opción para este proyecto.

3.2.4.2 Boot sequence

La secuencia de arranque en una Raspberry Pi es relativamente diferente a la de un sistema x86 tradicional [24]. En un dispositivo x86, la secuencia de arranque comienza con un POST [25]. POST son las siglas de Power On Self Test, que se encarga de realizar diversos procesos de diagnóstico y finaliza su función localizando un dispositivo arrancable en el sistema; este dispositivo puede ser desde un disco duro, un disco en estado sólido, una memoria externa USB, etc.

Existen dos tipos de programa para realizar este diagnóstico y configuración previa: uno de ellos es la BIOS (Basic Input Output System), que ahora es considerado el método de legado, y UEFI (Unified Extensible Firmware Interface). Sus diferencias principales residen en la metodología que usan para arrancar el sistema.

3.2.4.2.1 BIOS

Un sistema de legado que use BIOS realiza la secuencia POST y, tras eso, coloca al procesador en un estado de compatibilidad con sistemas operativos simples y antiguos llamado modo real, que funciona a dieciséis bits. En este modo, menos de un megabyte de memoria RAM

es accesible, y desaparece toda forma de paginación de memoria y de cualquier tipo de seguridad basada en hardware que el procesador pudiera soportar.

Para continuar con el arranque de un sistema operativo actual [26], BIOS busca todos los dispositivos conectados al sistema y, para cada uno de ellos realiza la siguiente secuencia de operaciones:

1. Se vuelcan los primeros 512 bytes del dispositivo en la memoria del sistema, concretamente en la dirección 0x7C00. Estos 512 primeros bytes con considerados el primer sector del dispositivo en cuestión.

2. Se localiza el penúltimo byte copiado. Si el valor de este byte es 0x55 continuar leyendo el siguiente byte, en caso contrario buscar el siguiente dispositivo y se repite este proceso.

3. Se localiza el último byte copiado. Si el valor de este byte es 0xAA se define el dispositivo como arrancable y se interpreta el valor de la dirección de memoria 0x7C00 como la primera instrucción a ejecutar. En caso contrario, se busca el siguiente dispositivo y se repite este proceso.

Suponiendo que las instrucciones situadas en 0x7C00 son las de un gestor de arranque, este se encarga de habilitar y configurar la Interrupt Descriptor Table (IDT) y la Global Descriptor Table (GDT), además de pasar el procesador de modo real a modo protegido y, si es necesario, habilitar el Long Mode Environment y la paginación de memoria 1:1 en algunas direcciones de esta [27].

El gestor de arranque es el encargado de cargar en la memoria el kernel del sistema operativo que se desea usar y ejecutarlo.

3.2.4.2.2 UEFI

En el caso de que se use UEFI, los pasos a seguir son sumamente similares, la única diferencia real es que el Firmware prescinde de cambiar a modo real y prepara todo para que cualquier ejecutable en una partición FAT (con un mapa de particiones GPT) pueda arrancar en modo protegido directamente. En función del procesador, el firmware también se encarga de habilitar la segmentación de memoria o la paginación de memoria 1:1 en diversas direcciones [28].

Cabe destacar que UEFI descarta el método del volcado del primer sector del disco en memoria y carga íntegramente todo el ejecutable que esté contenido en la partición FAT en una dirección de memoria determinada dinámicamente.

El ejecutable contenido en la partición FAT es el encargado de cargar en la memoria el kernel del sistema operativo que se desea usar y ejecutarlo.

3.2.4.2.3 Raspberry Boot Sequence

Como se ha mencionado anteriormente, los dispositivos Raspberry Pi utilizan un método de arranque que es sustancialmente diferente a UEFI y BIOS, y que además es único para el dispositivo.

Al encenderse, toda Raspberry Pi arranca con la CPU ARM en modo RESET, es por ese motivo que la GPU es la encargada de realizar el proceso de arranque.

Lo primero que realiza la GPU es cargar el First Stage Bootloader [24], que se encarga de cargar el Second Stage Bootloader en la memoria caché y lo ejecuta.

El Second Stage Bootloader se encarga de habilitar la SDRAM, carga el Third Stage Bootloader en la RAM, y lo ejecuta.

El Third Stage Bootloader se encarga de leer el firmware de la GPU (un archivo ELF), y este es el encargado de configurar los parámetros y dispositivos del sistema, además de cargar la imagen del kernel a ejecutar en memoria.

Una vez el kernel se encuentra en memoria, el firmware de la GPU cambia el estado de RESET de la CPU y esta comienza a ejecutar la primera instrucción del kernel, que probablemente sea la función main de este.

3.2.5 Linux Boot Sequence

El proceso de arranque de Linux es ligeramente distinto [29] a lo descrito anteriormente en BIOS; de hecho, también dista del proceso de arranque de Windows, a pesar de que ambos sistemas operativos deben enfrentarse a prácticamente las mismas fases al arrancar.

Cuando la BIOS cede el control de la ejecución al gestor de arranque, GRUB o LILO en Linux, este funciona siempre en modo real de 16 bits. Esto responsabiliza al Kernel a ejecutar todos los procedimientos necesarios para cambiar a modo protegido y habilitar la paginación de memoria.

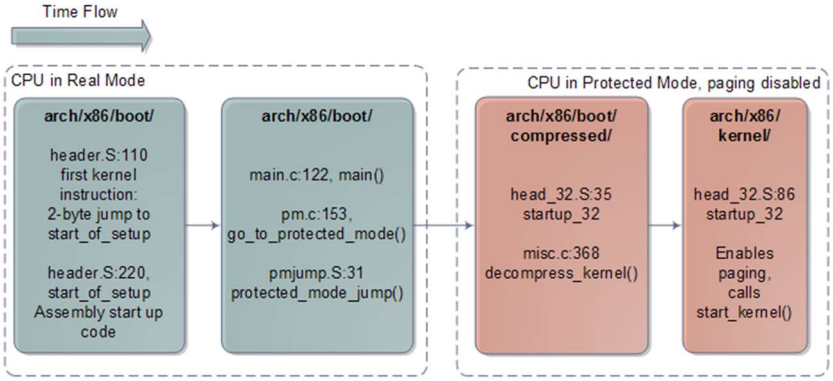

Ilustración 11 Inicialización del Kernel Linux. Fuente: [29]

En el caso de Windows, el proceso de arranque es muy similar al explicado anteriormente [30]. El gestor de arranque y winload se encargan de cambiar a modo protegido, habilitar la

paginación de memoria y configurar todo lo necesario para que el kernel NT pueda arrancar. Esto separa por completo, en dos ejecutables, las instrucciones que son procesadas en modo dieciséis bits y las que son procesadas en modo protegido de 32 o 64 bits.

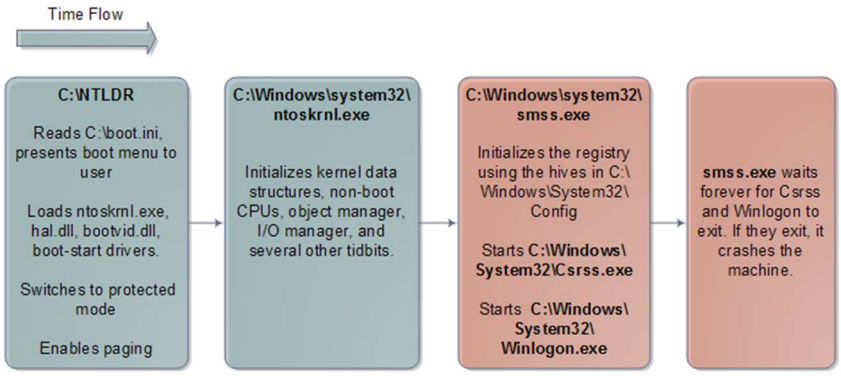

Ilustración 12 Inicialización del Kernel NT. Fuente: [29]

3.2.5.1 Descomprimir y configurar el Kernel Linux

Tal y como se ha mencionado anteriormente, el encargado de cargar el kernel Linux en memoria es el gestor de arranque, que puede ser tanto GRUB como LILO. Estos gestores de arranque, sin embargo, son demasiado grandes para ser almacenados íntegramente en el sector de arranque del disco. Para solventar este problema, se crea un pequeño gestor de arranque que es el encargado de cargar LILO o GRUB.

Este pequeño gestor de arranque se carga a sí mismo en la dirección 0x00007c00, y al ejecutarse se mueve a sí mismo a la dirección 0x00096a00. Una vez se ha realizado la operación, la primera parte del gestor de arranque designa una región de memoria que será usada como stack. Esta región de memoria está comprendida desde la dirección 0x00098000 hasta la dirección 0x000969ff.

Una vez se ha designado la memoria a usar como stack, el microgestor de arranque carga su segunda parte en la dirección de memoria 0x00096c00 y ejecuta el contenido de la dirección.

Con el gestor de arranque completamente cargado en RAM, se cargan los primeros 512 bytes de la imagen del Kernel en la dirección de memoria 0x00090000 y carga el resto de la imagen

en la dirección 0x00100000. Cuando el Kernel ya en memoria, el gestor de arranque ejecuta la función setup del Kernel.

La función setup del Kernel realiza la tarea de detectar y reinicializar todos los dispositivos del sistema. A pesar de que BIOS es capaz de enumerar e inicializar estos dispositivos, setup procede a realizar la misma función por motivos de portabilidad.

Una vez todos los dispositivos se encuentran listados y listos, la función setup procede a poner el procesador en modo protegido y habilita y configura la Interrupt Descriptor Table (IDT) y la Global Descriptor Table (GDT).

Por último, setup llama a la función startup_32.

Existen dos versiones de la función startup_32. La primera versión que se ejecuta tiene como objetivo descomprimir el núcleo Linux y ceder el control a la segunda versión, que se encuentra en el núcleo recién descomprimido.

Esta segunda versión es la encargada de preparar la IDT y la GDT para ser usadas, además de limpiar los flags del procesador. Una vez ha cumplido su cometido, esta función llama a start_kernel.

3.2.5.2 Ejecutar el Kernel Linux

start_kernel es responsable de completar la inicialización del núcleo Linux. Entre sus funciones más destacadas se destaca la ejecución del scheduler usando la función shed_init, la inicialización final de la IDT usando las funciones trap_init e init_IRQ, calcular la velocidad de reloj de la CPU usando calibrate_delay, inicializar los sistemas que controlan la fecha y la hora del sistema con time_init, y, por último, ejecutar /sbin/init.

3.2.5.3 SysVinit y systemd

Un init daemon es el primer proceso (PID 1) que ejecuta el sistema y que se encarga de arrancar y gestionar todos los servicios y daemons del sistema. Tanto SysVinit como systemd entran dentro de esta categoría de programas.

En la mayoría de las distribuciones Linux es posible encontrar el uno o el otro en /sbin/init, a pesar de que esta es la localización clásica de SysVinit. Para evitar conflictos, systemd se encuentra en otra ruta diferente, pero si es el init daemon de la distribución se le hace un symlink a /sbin/init. De esta forma, sin importar cual sea el init daemon escogido, el kernel Linux siempre accede a la misma ubicación para ejecutarlo.

3.2.5.4 Booting Linux en ARM

A pesar de que el proceso de arranque descrito anteriormente es exclusivo de las máquinas x86 y x86_64; el proceso en la arquitectura es esencialmente el mismo. Las direcciones de memoria y los modos del procesador cambian a los necesarios para el arranque de un procesador con arquitectura ARM, pero el procedimiento general es muy parecido.

Las variaciones principales se encuentran antes de start_kernel, a partir de este punto el código es exactamente el mismo en todas las plataformas.

Las instrucciones específicas de ARM consisten en la búsqueda e identificación del tipo de procesador, la búsqueda e identificación del tipo de máquina, la creación de las páginas de memoria en la tabla de la MMU, cambiar el registro l3 para habilitar la MMU, realizar un flush al procesador, y, por último, habilitar la MMU.

Capítulo 4: Análisis de referentes

Una vez definidos los objetivos y requisitos del proyecto, comenzó una búsqueda de software que pudiera actuar como referente. A pesar de que diversas alternativas surgieron, solo una pudo compararse con todo lo esperado del proyecto: NoMachine.

En cuanto a los referentes a nivel teórico, se ha encontrado una gran variedad de libros que tratan acerca de las tecnologías mencionadas, pero pocos de ellos mencionan la arquitectura ARM, y es posible que esa falta de mención, y por tanto de documentación específica, pueda desembocar en una exhaustiva investigación de fuentes no fiables y en la realización de diversas pruebas, que puede ralentizar el avance del desarrollo.

4.1 NoMachine

NoMachine es un software de escritorio remoto gratuito para uso personal y que contempla prácticamente todas las características mencionadas en la toma de requisitos de este proyecto. Sin embargo, este software es completamente propietario y su licencia restrictiva para el uso comercial hace que una solución similar de código abierto sea aún más atractiva.

A pesar de que NoMachine cuente con software para Raspberry Pi, es posible que el software no se ejecute correctamente en caso de que la aceleración por hardware del sistema destino, o del sistema origen, no estén habilitadas. Esto puede resultar un problema en el caso de máquinas virtuales sin aceleración gráfica disponible.

Capítulo V: Metodología

Se contempla el uso de tres metodologías diferentes según la tarea a desempeñar en este Trabajo de Fin de Grado: la primera metodología es usada para la búsqueda y síntesis de información, la segunda es usada durante el desarrollo, y la tercera es usada durante el testeo del sistema.

5.1 Metodología de búsqueda de información

La metodología escogida para la búsqueda y síntesis de información radica en una primera búsqueda en profundidad a través de un motor de búsqueda convencional. En esta búsqueda se contemplan todos los resultados que tienen fuentes consideradas como válidas; es decir, fuentes que provienen de estudios o artículos de fuentes fiables. Además, se investiga la fuente y el autor para revisar si el artículo o el estudio es fiable.

Tras obtener toda la información posible de las fuentes fiables, se procede a contrastar esa información con información de carácter fiable que ponga en duda los estudios encontrados. Una vez se ha considerado ambos lados de la información se realiza una síntesis de esta.

Una vez se han agotado las fuentes de información fiables, se procede a la búsqueda de información no fiable en foros especializados. Una vez obtenida esta información se contrasta tratando de replicar la forma en que han obtenido la información. En caso de que el experimento realizado para replicar sea concluyente y arroje los mismos resultados que están descritos en el foro, se considera esa información como información fiable.

Por último, se buscan e investigan libros ya publicados que traten de temas circundantes al tema del que se busca información. Una vez se ha realizado una pequeña lista de libros, se procede a leer las secciones de mayor relevancia y se sintetiza la información obtenida.

5.2 Metodología de desarrollo

Para desarrollar este sistema se ha optado por usar la clásica metodología waterfall, muy típica de la década de los 90. Esta metodología se compone de 4 fases: toma de requisitos, diseño, implementación, y testeo.

5.2.1 Toma de requisitos

En esta fase se elabora una lista de requisitos que el sistema debe cumplir para estar considerado como acabado. Esta lista de requisitos queda expuesta en el siguiente capítulo de este mismo documento.

5.2.2 Diseño

En esta fase se elaboran todos los documentos necesarios para el diseño y la preproducción del sistema. De estos diseños se comenzará a elaborar el producto final. Estos diseños parten de la toma de requisitos.

5.2.3 Implementación

En esta etapa se implementa el software y se construye la solución según los documentos de diseño elaborados en la fase anterior. Esta es la etapa más larga del proyecto que culmina con la creación del sistema y su respectiva documentación.

5.2.4 Testeo

En esta última etapa del desarrollo se testea el software para asegurar que cumple con los requisitos y con el comportamiento esperado. En caso de que alguna sección del software no pase la prueba, esta es enviada de nuevo a la etapa de implementación.

Capítulo VI: Diseño de la solución

Se contempla una división de los requisitos del sistema en dos categorías. La primera categoría concentra todos los requisitos funcionales del proyecto, la segunda categoría concentra los requisitos tecnológicos del sistema.

También se ha optado por realizar dos listas de requisitos por cada categoría: en la primera lista se encuentran los requisitos indispensables para que el sistema cumpla con los objetivos descritos en el capítulo tres; en la segunda se encuentran una serie de requisitos que el producto debería incluir a medio o largo plazo, pero cuya compleción no está garantizada en el alcance de este trabajo.

6.1 Requisitos Funcionales

Los requisitos funcionales definen el comportamiento de un software o sistema. En este caso, los requisitos detectados son los siguientes.

6.1.1 Requisitos funcionales primarios

- Proveer una imagen del conector actualizada.
- Descargar una imagen del conector desde un servidor TFTP.
- Arrancar una imagen descargada sin necesidad de Tarjeta microSD.
- Iniciar sesión.
- Proveer una máquina virtual no usada.
- Conectar con la máquina virtual.
- Transmitir la entrada de teclado y ratón a la máquina virtual.
- Monitorizar el estado de la máquina virtual.

6.1.2 Requisitos funcionales secundarios

- Transmitir dispositivos USB genéricos a la máquina virtual.
- Permitir más de un monitor.

Además de estos requisitos funcionales, también se han detectado los siguientes requisitos no funcionales que serán tratados como requisitos secundarios.

- Arrancar un dispositivo debe ser rápido.
- Cambiar un dispositivo debe ser breve.
- Iniciar sesión debe ser intuitivo.
- Administrar las máquinas virtuales debe ser simple.
- Administrar más de un servidor debe ser rápido.
- Administrar más de un servidor debe ser sencillo.
- Usar el dispositivo no debe implicar el uso de la consola de comandos.

6.2 Requisitos Tecnológicos

Los requisitos tecnológicos definen qué es necesario, a nivel de tecnología, para llevar a cabo un proyecto. En este caso, son los siguientes.

6.2.1 Requisitos tecnológicos primarios

- Disponer de un servidor con la capacidad de virtualización asistida por hardware (AMD-V o Intel VT-x).
- Disponer de un hipervisor.
- Disponer de varios dispositivos Raspberry Pi.
- Disponer de un SDK para compilar o modificar una distribución Linux.
- Disponer de un SDK compatible con ARM o ARM64.
- Disponer de una instalación de red Ethernet.

6.2.2 Requisitos tecnológicos secundarios

- Utilizar lenguajes de programación eficientes para mejorar el rendimiento del sistema.
- Ofrecer un sistema similar a relay agents para los routers.
- Distribuir el mayor número de componentes del sistema mediante Docker.

Mediante todos estos requisitos, tanto funcionales como tecnológicos, y los objetivos mencionados anteriormente; queda definido tanto el alcance del proyecto como sus características significativas.

6.3 Diseño de la solución

Dadas las anteriores características, se ha procedido a realizar el diseño de la solución final a tres niveles diferentes. El primero de todos es el nivel físico, que muestra a nivel de hardware qué es lo necesario para llevar a cabo el proyecto y cómo están interconectados los diferentes componentes; el segundo es el esquema a nivel lógico, que muestra las interconexiones lógicas de los diferentes componentes de la solución; el tercer nivel es la arquitectura de red, que muestra un esquema del estado final de la red.

6.3.1 Diseño físico

A nivel físico, se ha optado por contar con un servidor Dell R710 como dispositivo central de la solución. Entorno a este dispositivo, se ha construido una infraestructura que permita que los clientes, tanto Raspberry Pis como otros dispositivos, puedan conectarse a la red y a internet, además de acceder a los recursos compartidos de esta. Para lograr esto, se ha optado por adquirir un switch y contar con el router del ISP para garantizar el acceso a Internet.

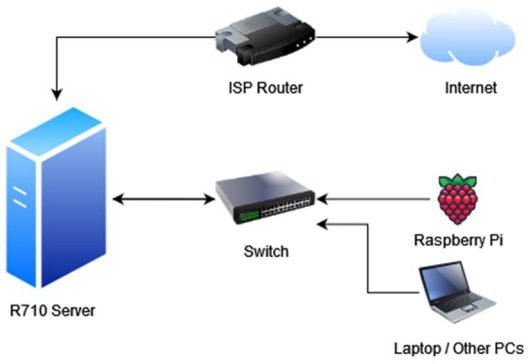

Ilustración 13 Diagrama físico. Fuente: propia.

6.3.2 Diseño lógico

El diseño lógico de la solución se centra en los distintos componentes conceptuales, virtuales o no, que permiten el correcto funcionamiento del proyecto.

Basándose en el diseño físico, para este proyecto se ha optado por virtualizar una instancia de pfSense que se encarga de ejercer como puerta de enlace predeterminada hacia Internet. Además de actuar como router, también constituye el núcleo de la red y provee a ésta de servicios como DHCP, DNS, etc.

Esta instancia de pfSense está conectada, también, a un switch externo, que es el encargado de ofrecer conectividad tanto al propio servidor R710 como al resto de dispositivos que quieran estar conectados a la red. En el caso de que se requieran de puntos de acceso WIFI, estos serían conectados, también, a este switch.

Como se ha mencionado en el párrafo anterior, el switch es el encargado de conectar el servidor R710 a internet y al resto de la red. Esto es debido a que tanto el puerto bce0 como el puerto bce1 son pasados directamente a la máquina virtual de pfSense. Esta decisión ha sido tomada para minimizar las latencias y mejorar el rendimiento total de la red. Es posible realizar el switching de forma virtual dentro del propio servidor mediante VirtIO; pero la latencia aumenta, la carga de la CPU es superior, y el rendimiento de la red baja.

La interfaz en01 provee al bridge de VirtIO de conexión a la red, y este bridge es el encargado de retransmitir los paquetes a las máquinas virtuales y los contenedores Docker que lo usan.

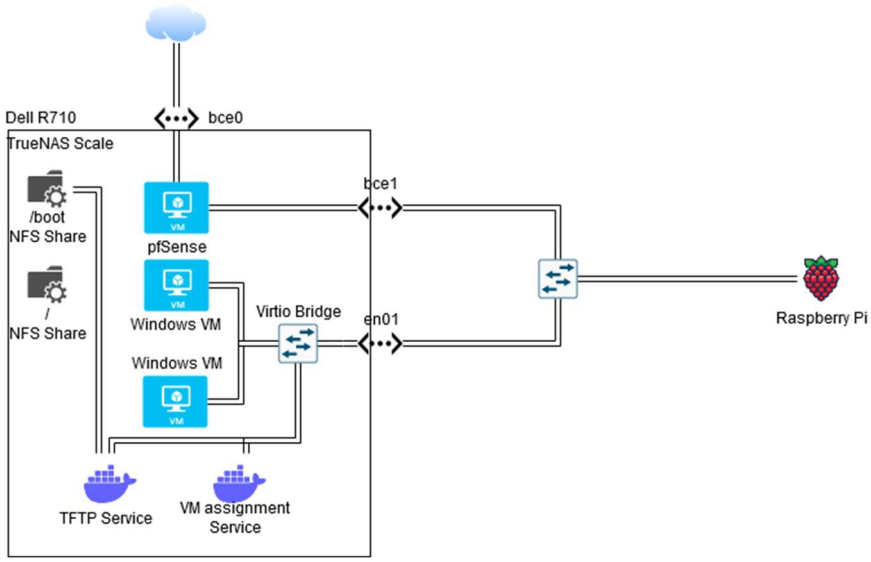

Ilustración 14 Diagrama lógico. Fuente: propia.

6.3.3 Diseño de red

El diseño de red consta de un diagrama simplificado en el que se especifican las direcciones de red de cada componente del sistema. Se omite por completo la existencia del servicio DHCP, dejando implícito que las direcciones asignadas a partir de la dirección 10.0.1.100 son todas asignadas de forma dinámica.

Como se puede observar en el diagrama, la solución precisa de dos redes. La primera es constituida entre el router de la operadora y la máquina pfSense. Esta red podría ser obviada si la máquina virtual pfSense se encargase de establecer la conexión (probablemente PPPoE) a internet. La segunda red consiste en la red de área local de la solución y, por ende, de la propia empresa.

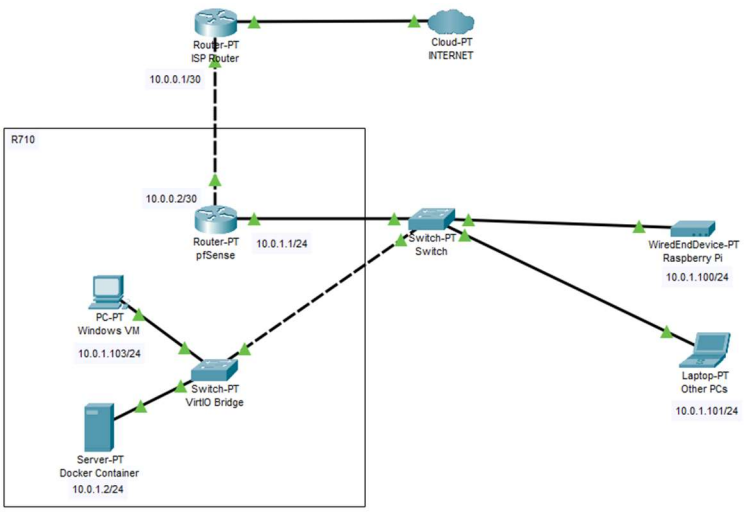

Ilustración 15 Diagrama de Red. Fuente: propia.

6.3.4 Detalles de la solución

La elección de software para este proyecto se ha hecho llevando a cabo dos criterios: el primero consiste en encontrar un equilibrio entre rendimiento, consumo energético, y precio; el segundo criterio consiste en la practicidad de la solución. Por motivos de practicidad, se ha supuesto que esta solución se va a implementar en una mediana empresa; sin embargo, con el diseño actual es posible trasladar la solución a empresas de mayor tamaño haciendo ligeros cambios en el software y externalizando ciertos servicios que otorga el nodo principal.

Se ha considerado usar Proxmox VE como hipervisor sobre el que ejecutar las máquinas virtuales, pero eso requería virtualizar un servicio de NAS y realizar un passthrough de los discos hacia la máquina virtual donde residiría el NAS. Otra solución requeriría tener una máquina externa dedicada exclusivamente a ofrecer servicios de NAS.

Para simplificar el proyecto y ofrecer la máxima cantidad posible de servicios, se ha optado por usar TrueNAS Scale como hipervisor, que además ofrece servicios de NAS.

A nivel de rendimiento, esta decisión no ha tenido ningún impacto con respecto al uso de software más especializado, como puede ser Proxmox VE, y además este ofrece características similares en cuanto a la gestión de clústeres. Este nulo impacto es debido a que el hipervisor con el que trabajan ambos sistemas operativos es el mismo: KVM. Como se pudo observar, KVM es el hipervisor de código abierto que mejor rinde en condiciones donde una cantidad considerable de máquinas virtuales realizan tareas simples.

En cuanto a la virtualización de dispositivos de red, se ha optado por virtualizar pfSense: un firewall con funciones de router completamente open source y basado en BSD. La decisión respecto a la instalación de este software reside en su reputación como “World's Most Trusted Open Source Firewall [31]” y en su amplia documentación y comunidad.

Capítulo VII: Desarrollo del proyecto

El desarrollo del proyecto está concebido en tres fases secuenciales. La primera tiene como objetivo principal conseguir arrancar una imagen cualquiera mediante Network Boot en una Raspberry Pi; la segunda tiene como objetivo principal modificar una imagen Linux y embeberle un software de escritorio remoto; la tercera tiene como objetivo principal automatizar el proceso de conexión y asignación a las máquinas virtuales. Estas tres fases están desglosadas en el conjunto de tareas mostrado a continuación.

7.1 Descarga e instalación de TrueNAS Scale

TrueNAS Scale es una solución de código abierto para estructuras hiperconvergentes, donde se virtualiza, mediante el uso de un hipervisor, los elementos clásicos de la infraestructura (como puede ser el sistema de almacenamiento o el sistema de red), desarrollado por iXSystems.

Para descargar este software es necesario contar con una memoria USB con, al menos, cuatro gigabytes de capacidad.

Se procede a descargar TrueNAS Scale desde su página oficial de descargas (*https://www.truenas.com/download-truenas-scale/*), donde se solicita algún tipo de identificación, ya sea mediante "Continue with Google", "Continue with Github", o "Continue with Facebook". Una vez se ha procedido a la autentificación, la web redirige al usuario a la página de descargas, donde se puede descargar la imagen ISO del software.

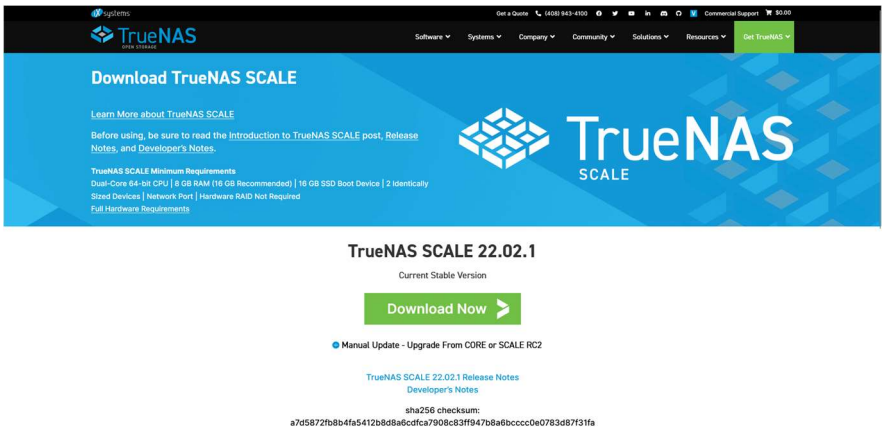

Ilustración 16 Página descarga TrueNAS Scale. Fuente: propia.

Es necesario quemar la imagen ISO descargada en la memoria USB antes mencionada, para este propósito se usa la herramienta Rufus. Rufus se puede descargar desde su página principal (*https://rufus.ie/*), donde se explican algunas de sus características principales. Para proceder a la descarga, se debe desplazar la página hacia abajo hasta alcanzar la sección de Descargas, donde se descarga la versión portable de la herramienta.

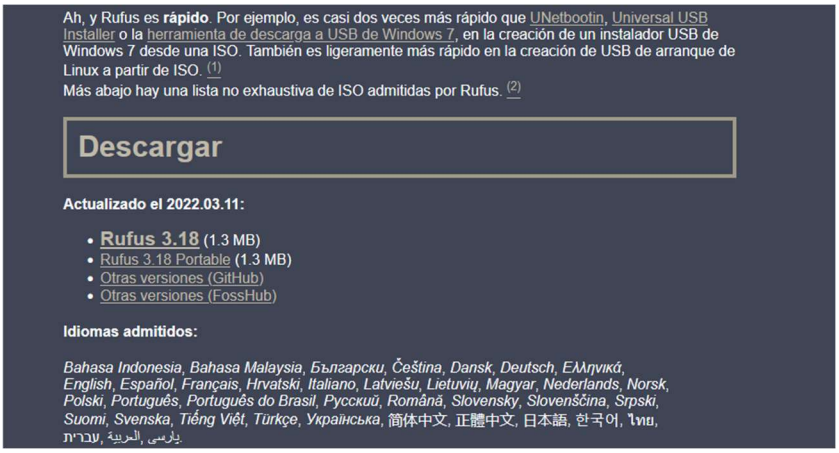

Ilustración 17 Página de descarga de Rufus. Fuente: propia.

Al ejecutar Rufus, se muestra una interfaz con todos los campos necesarios para quemar cualquier imagen en una memoria USB. Se debe seleccionar el dispositivo donde se pretende quemar la imagen, el archivo que contiene la imagen a quemar, y el resto de los campos son rellenados de forma automática. En caso contrario, es posible que la imagen que se esté tratando de quemar no sea la correcta, o que Rufus sea incapaz de detectar el tipo de sistema operativo que se pretende volcar en la memoria USB. En cualquier caso, se recomienda volver a descargar la imagen y reiniciar Rufus. Si sigue sin completar el resto de los campos, se recomienda usar un esquema de partición MBR y configurar el sistema destino como BIOS o UEFI.

Respecto a las opciones de formateo, se recomienda usar FAT32 y un tamaño de clúster no superior a 32 kilobytes.

Una vez configurada la utilidad, solo hay que hacer clic en empezar y la unidad destino será poblada con la imagen de disco autoarrancable.

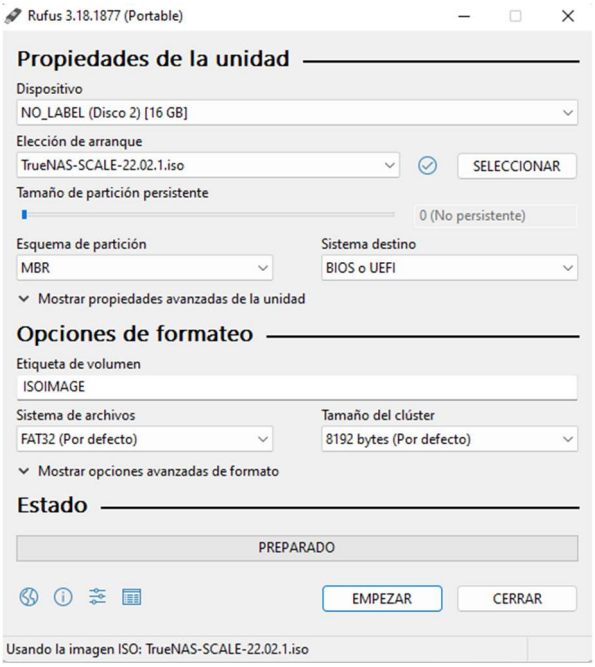

Ilustración 18 Interfaz de Rufus. Fuente: propia.

7.1.1 Preparación de IDRAC 6

Antes de proceder con la instalación de TrueNAS Scale, es conveniente configurar IDRAC 6; el módulo integrado de acceso remoto de los servidores Dell Poweredge. El software de la versión seis de este dispositivo es ligeramente antiguo, data de 2018, y debido a esta antigüedad hay que realizar diversas configuraciones antes de proceder a usarlo.

El primer paso por realizar es conectar el puerto de administración del servidor Dell R710 a la red local donde se esté trabajando. Una vez la conexión sea exitosa, se procede a buscar el módulo frontal del IDRAC en el servidor.

Ilustración 19 Panel frontal IDRAC 6. Fuente [33].

En este módulo, se navegará hasta Setup => DRAC => Static IP y se procede a configurar la dirección IP de la unidad IDRAC, la máscara de subred de la red, y la puerta de enlace

predeterminada. Con esta configuración debería ser posible acceder a la interfaz web de configuración.

Es posible que, al intentar acceder a la página de administración web del IDRAC (que reside en la IP que se ha especificado anteriormente), el navegador muestre un error en el certificado SSL. Se debe ignorar el error y proceder a la página con normalidad.

La página de bienvenida de IDRAC es una página de inicio de sesión en el dispositivo. Las credenciales por defecto son:

- Usuario: root
- Contraseña: calvin

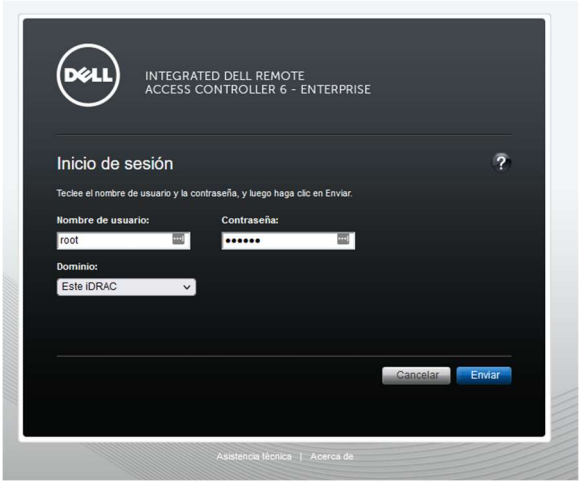

Ilustración 20 Inicio de sesión IDRAC 6. Fuente: propia.

Una vez se ha logrado iniciar sesión, el sistema muestra una ventana de estado, donde se muestra el estado de los diferentes subsistemas del servidor. A la derecha se encuentra, también, una vista previa de la consola virtual, y un botón para acceder a ella.

La consola virtual de IDRAC 6 es un archivo JNLP (Java Network Launching Protocol), que se debe abrir con el binario javaws localizado en la carpeta de binarios de una instalación normal de JRE (Java Runtime Environment). Sin embargo, la ejecución de dicho archivo se verá interrumpida por un error de seguridad.

Para solucionar dicho error, se debe acceder al panel de control de java y modificar las excepciones de seguridad.

Para ello, se debe abrir la aplicación Configurar Java, que se instala con JRE, y navegar hasta la pestaña Seguridad. Allí se encuentra la lista de excepciones de sitios, donde se debe añadir la dirección IP de IDRAC tanto con el protocolo HTTP como con el protocolo HTTPS.

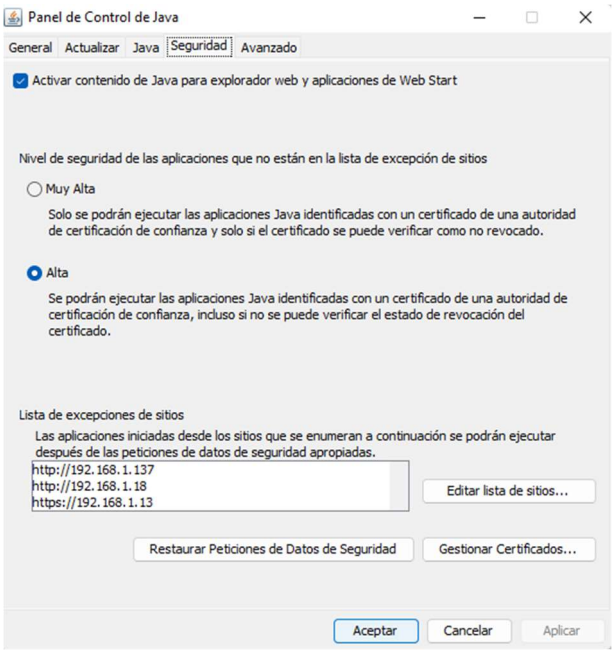

Ilustración 21 Panel de Control de Java. Fuente: propia.

Sin embargo, el añadir estas excepciones de seguridad no es suficiente para el correcto funcionamiento de la consola virtual.

Para que Java sea capaz de conectarse a IDRAC es necesario modificar el archivo java.security, que se encuentra en la carpeta de librerías de la instalación de JRE, bajo el directorio security.

En el archivo, se debe buscar la línea “jdk.tls.disabledAlgorithms”, donde se debe borrar el algoritmo RC4 de la enumeración.

```
# Example:
#   jdk.tls.disabledAlgorithms=MD5, SSLv3, DSA, RSA keySize < 2048
#jdk.tls.disabledAlgorithms=SSLv3, RC4, DES, MD5withRSA, DH keySize < 1024, \
#   EC keySize < 224, 3DES_EDE_CBC, anon, NULL
jdk.tls.disabledAlgorithms=SSLv3, DES, MD5withRSA, DH keySize < 1024, \
    EC keySize < 224, 3DES_EDE_CBC, anon, NULL
```

Ilustración 22 jave.security. Fuente: propia.

Una vez borrado el algoritmo, se debe guardar el archivo y se puede lanzar la consola virtual.

Es completamente normal que, durante el inicio de la consola virtual, Java notifique al usuario que existe un riesgo de seguridad elevado al continuar con la ejecución de la aplicación. Esto es debido a que hace uso de un algoritmo considerado como inseguro para los estándares de seguridad actuales. En este caso, se deben ignorar las advertencias y continuar con la ejecución del programa.

La pantalla inicial de la consola virtual recibe al usuario con una ventana totalmente en negro y una barra de herramientas. Para proceder con la configuración se debe navegar hasta Alimentación, y seleccionar la opción “Encender el sistema”.

7.1.2 Configuración de BIOS Dell R710

Para acceder a la BIOS en el servidor Dell R710, se debe pulsar la tecla F2 durante el arranque del sistema. Al hacerlo correctamente, un mensaje en la parte superior derecha aparece.

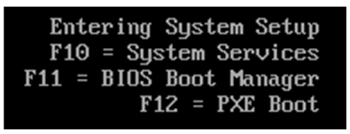

Ilustración 23 Entering BIOS Dell R710. Fuente: propia.

Pasados un par de minutos, la ventana de bienvenida de la BIOS es mostrada al usuario. En esta pantalla se procede a modificar la fecha y la hora del sistema para adaptarla a la actual. Tras modificar estos valores se procede a navegar hasta SATA Settings, donde se debe colocar el modo del controlador SATA en modo ATA y no en modo IDE.

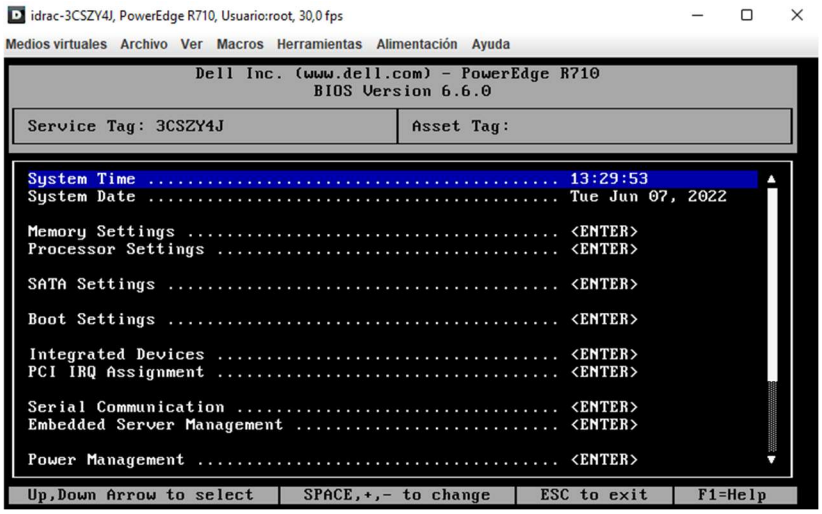

Ilustración 24 BIOS Dell R710. Fuente: propia.

También se debe navegar a Integrated Devices y habilitar el controlador RAID integrado, ya que sin este es imposible utilizar las ranuras para almacenamiento posteriores.

Una vez realizada esta configuración, se procede a salir de BIOS pulsando ESC y seleccionando la opción que guarda todos los cambios y reinicia el dispositivo. Tras reiniciar, se procede a observar con atención el proceso de arranque, ya que cuando se haga mención del controlador RAID se debe entrar en su utilidad para configurarlo.

En esta situación concreta, no se va a realizar ninguna configuración especial de RAID, y lo que se requiere es la creación de un volumen virtual para cada volumen físico. Esto se consigue marcando que se quiere usar un volumen tipo RAID 0 y entregando un solo dispositivo al configurador. Esto permite que sea TrueNAS Scale quien gestione los discos y su correspondiente paridad, en caso de querer contar con ella.

La configuración debe quedar de la siguiente forma, aunque depende íntegramente del número de discos que el sistema posea.

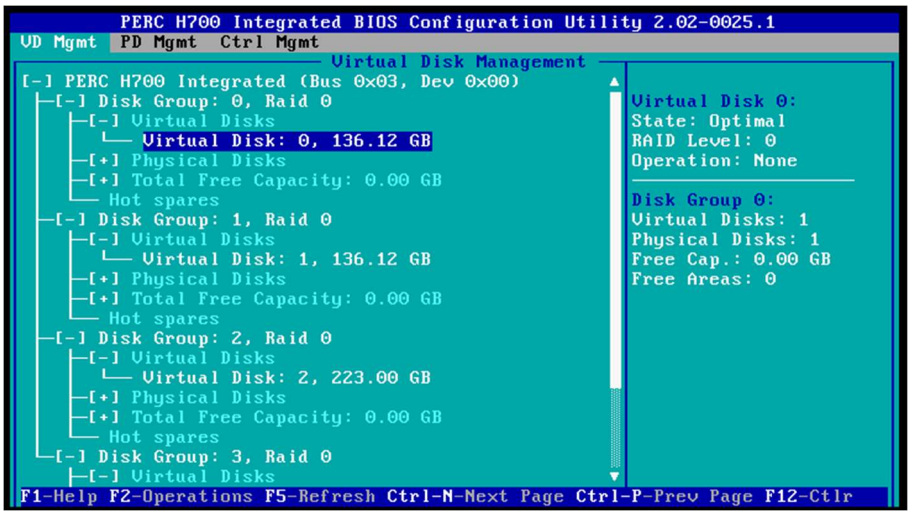

Ilustración 25 PERC H700 Configuration Utility. Fuente: propia.

7.1.3 Instalación de TrueNAS Scale

Al insertar el medio de instalación de TrueNAS Scale en el servidor, este da la bienvenida al usuario con la pantalla de instalación de TrueNAS Scale. En esta pantalla, se seleccionará la opción Install/Upgrade.

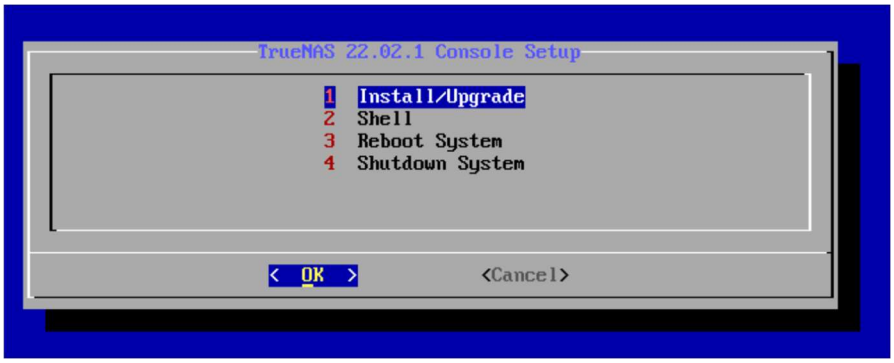

Ilustración 26 Instalación TrueNAS Scale. Fuente: propia.

A continuación, se muestra la lista de discos a utilizar para la instalación del sistema operativo. Estos discos, solo pueden hospedar al sistema y sus herramientas, y no pueden ser utilizados como almacenamiento para máquinas virtuales.

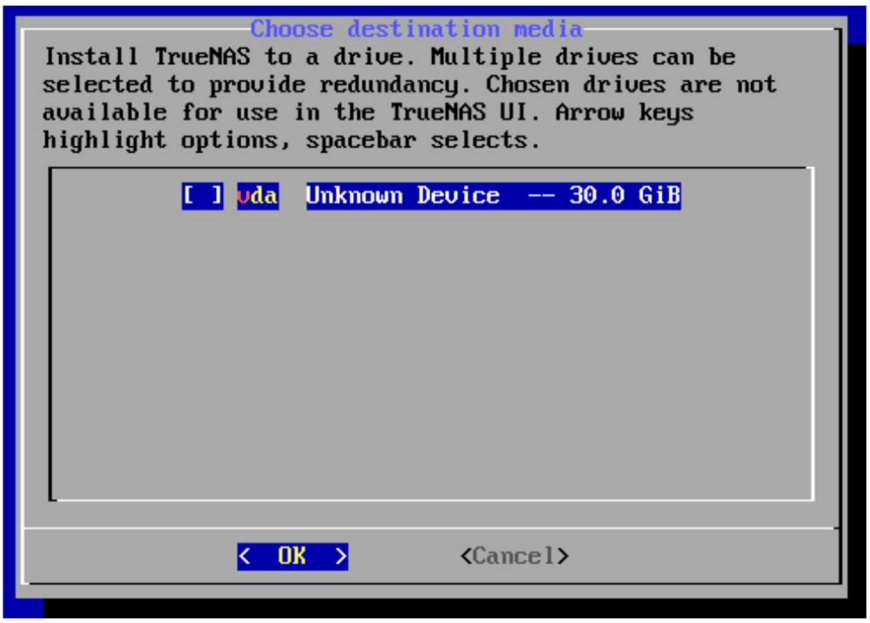

Ilustración 27 Selección de Discos TrueNAS Scale. Fuente: propia.

Tras seleccionar los discos deseados, se procede a la ventana de selección de contraseña para el usuario root.

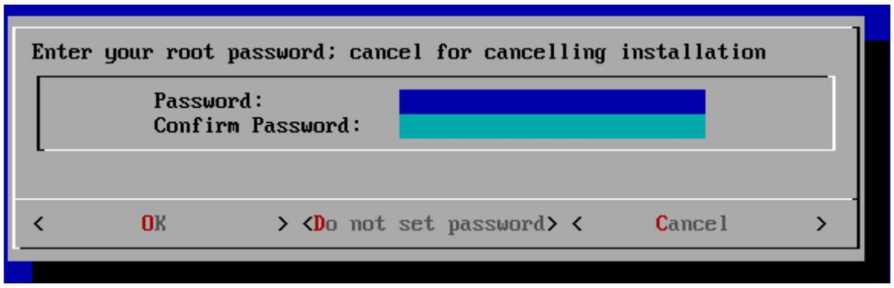

Ilustración 28 Ventana selección contraseña TrueNAS Scale. Fuente: propia.

Con la contraseña para el usuario maestro introducida, el asistente de instalación comienza a instalar TrueNas Scale en los discos seleccionados.

Es posible que la instalación falle con un error de disco desconocido, si este es el caso se debe realizar un formato de bajo nivel a los discos y reiniciar el proceso de instalación. Se desconoce el motivo por el que se produce este error, pero esta ha sido la única solución que se ha encontrado.

En caso de que la instalación se complete sin errores, TrueNAS muestra la siguiente ventana.

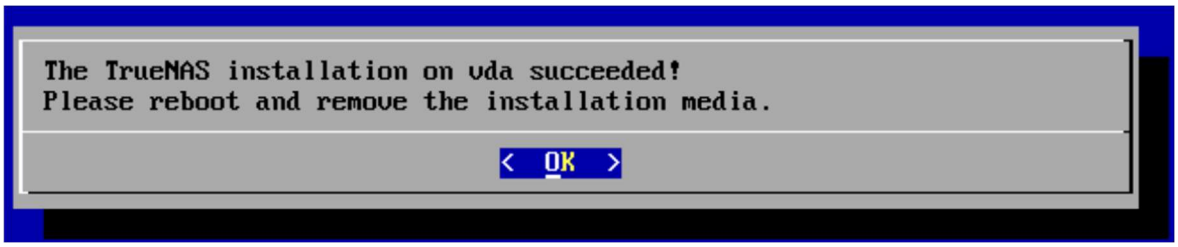

Ilustración 29 Instalación completada TrueNAS Scale. Fuente: propia.

7.2 Configuración TrueNAS Scale

La primera vez que TrueNAS Scale se inicia, se muestra el usuario una pantalla en la que se le permite realizar diferentes configuraciones: configurar las interfaces de red, configurar los ajustes de red, configurar las rutas estáticas, etc.

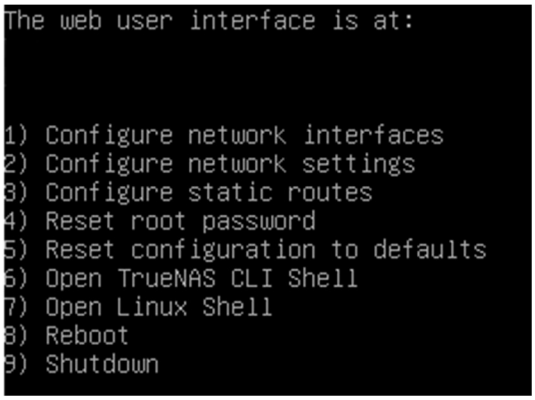

Ilustración 30 Pantalla principal TrueNAS Scale. Fuente: propia.

También se indica en qué dirección IP se encuentra la interfaz del configurador web. Lo más probable es que TrueNAS no haya sido capaz de detectar automáticamente la configuración y se deba realizar una configuración inicial a mano. Para ello, se debe entrar a la configuración de interfaces de red y seleccionar la interfaz que esté conectada a la red local. Es importante tener en cuenta que no se deben usar las interfaces de la tarjeta de red PCI-Express, debe ser una de las 4 interfaces del NIC.

En esta pantalla de configuración se seleccionará el apartado ipv4_dhcp y se debe activarlo, dejando la configuración como en la siguiente ilustración.

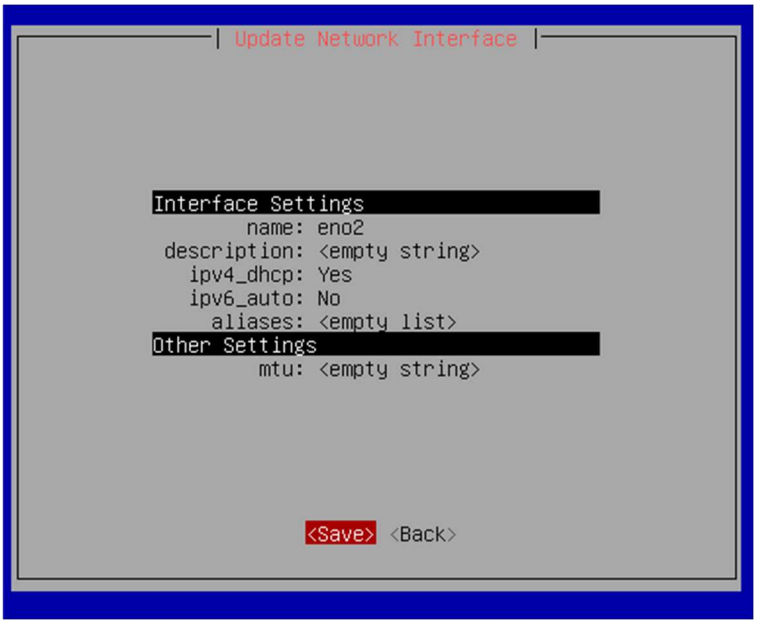

Ilustración 31 Ejemplo de configuración de la interfaz. Fuente: propia.

Una vez se ha guardado la configuración, en la pantalla principal, debe aparecer una dirección IP en la que está alojado el configurador web.

Si se accede al configurador web, el usuario es bienvenido con la pantalla principal de TrueNAS Scale, donde se muestran ciertos elementos importantes del estado del servidor.

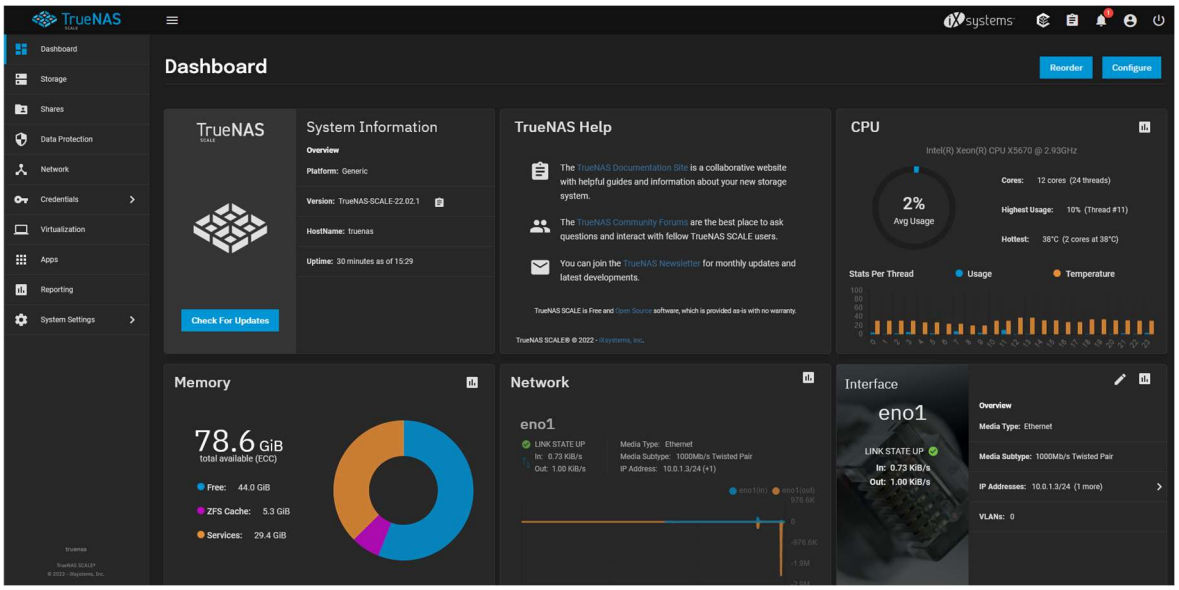

Ilustración 32 Dashboard TrueNAS Scale. Fuente: propia.

La siguiente tarea que se debe realizar, es configurar la Pool de discos donde se van a almacenar todos los datos necesarios para el sistema, además de albergar las imágenes de disco, etc.

Esto se consigue navegando a la sección Storage del menú de la izquierda del Dashboard y creando una pool de discos nueva.

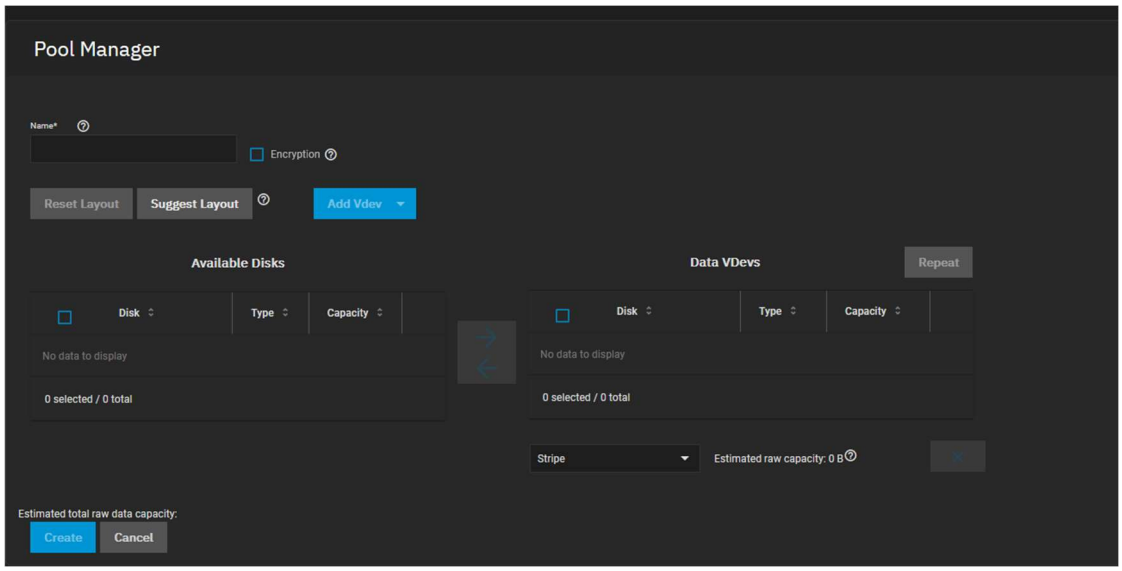

Ilustración 33 Creación de una pool de discos. Fuente: propia.

En configuración de la pool, se introduce el nombre Pool1 y se seleccionan los discos que quieran ser parte de la pool. Tras eso, se confirma la creación y la configuración inicial de TrueNAS Scale queda terminada.

7.2.1 Configuración NFS

Para configurar un share NFS en TrueNAS Scale, se debe acceder a la sección Shares del panel izquierdo. En esta sección, se encuentran los diferentes servicios de almacenamiento en red que soporta el software. En este caso concreto, se requiere el uso de NFS. Para acceder a la configuración del módulo NFS, el usuario debe pulsar sobre Unix (NFS) Shares. Esto lo llevará a una página nueva, donde podrá añadir un share nuevo.

Para este proyecto se requieren dos shares: el primero, es una carpeta llamada images, que se creará dentro de la Pool1; y el segundo es una carpeta llamada NFS, que se creará en la misma pool.

Una vez la configuración se haya creado correctamente, el sistema mostrará el siguiente estado en la página Shares.

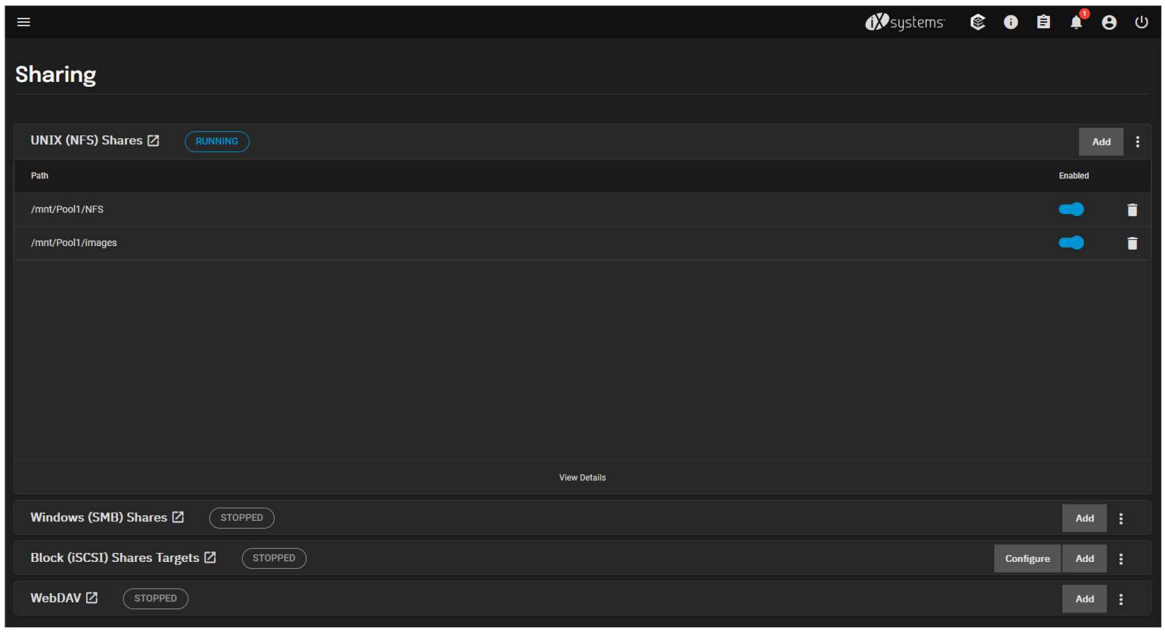

Ilustración 34 Shares TrueNAS Scale. Fuente: propia.

Con esta configuración es posible continuar con el proyecto; sin embargo, para que el funcionamiento del arranque en red funcione correctamente (y no produzca errores extraños), se debe modificar el fichero /etc/export, dejándolo con los siguientes parámetros.

```
 GNU nano 5.4                          /etc/exports


"/mnt/Pool1/NFS"\
        *(sec=sys,rw,anonuid=0,anongid=0,all_squash,insecure,subtree_check)
"/mnt/Pool1/images"\
        10.0.1.0/24(sec=sys,rw,anonuid=0,anongid=0,all_squash,insecure,subtree_check)
```

Ilustración 35 /etc/exports. Fuente: propia.

7.3 Crear Máquinas Virtuales

Crear máquinas virtuales en TrueNAS Scale es un proceso sencillo. Se debe navegar hasta la sección Virtualización. Allí, se debe entrar en el menú Añadir, que muestra un panel lateral en el que un asistente guía al usuario para realizar toda la configuración necesaria.

Para la configuración de una máquina virtual basada en Windows, se debe configurar de la siguiente forma:

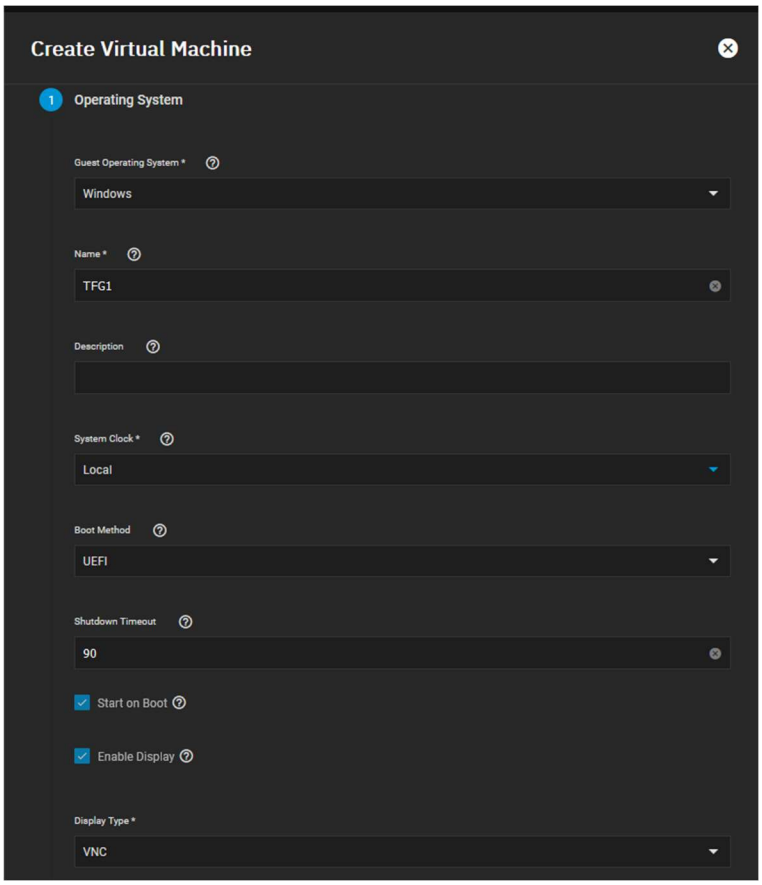

Ilustración 36 Create Virtual Machine 1. Fuente: propia.

Esta configuración indica a KVM que el sistema operativo a hospedar es un sistema Windows que espera arrancar con UEFI (por lo que debe usar una implementación de TianoCore). Además, también especifica que el tipo de pantalla a utilizar con esta máquina virtual es una pantalla virtual VNC, lo que permite al usuario conectarse mediante una dirección IP.

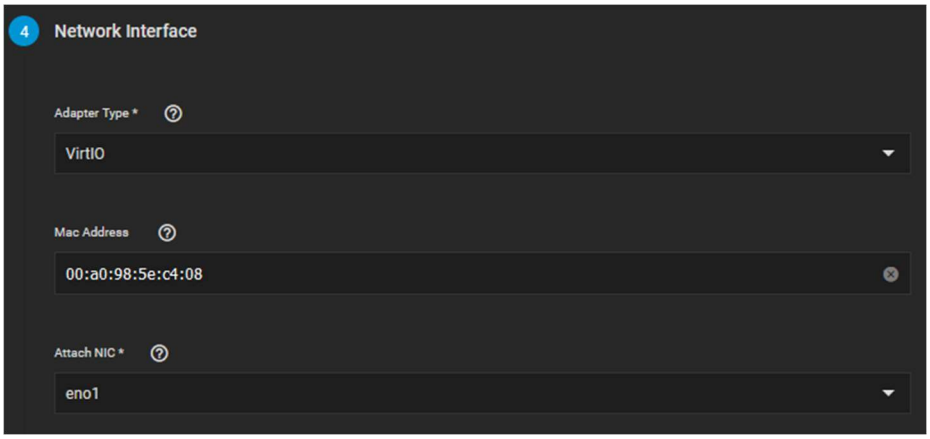

Ilustración 37 Create Virtual Machine. Fuente: propia.

En la sección Interfaces de red, se debe seleccionar el tipo de adaptador como “VirtIO”, ya que es el que ofrece mejor rendimiento, y se debe seleccionar eno1 como “Attach NIC”.

Con esta configuración inicial, ya es posible arrancar la máquina virtual e instalar el sistema operativo a hospedar.

7.3.1 Instalar el servicio en Windows

Para poder instalar el servicio de heartbeat en Windows hay que recurrir a una herramienta de código abierto llamada NSSM [34]. Esta herramienta es un pequeño gestor de servicios que permite instalar un servicio en Windows sin depender del gestor de servicios integrado.

Para usarlo, hay que descargar la última versión del programa y abrirlo mediante una consola de comandos con el siguiente comando:

```
1.  nssm.exe install Heartbeat_Service_NSSM
```

Al ejecutar el comando, un dialogo del sistema aparece. En este dialogo se debe indicar la ubicación del archivo (en este caso el archivo Heartbeat Service.exe). Con la ubicación indicada, se puede hacer clic en instalar y el servicio quedará instalado.

7.3.2 Instalar y configurar pfSense

La sección más crítica de la instalación y configuración de pfSense reside en la configuración de la máquina virtual, a la que hay que añadir una entrada para hacer un passthrough de la tarjeta de red PCI-Express.

Para localizar el dispositivo, se debe ejecutar el siguiente comando y buscar en el PCI-Express tree el identificador requerido después en el configurador de TrueNAS Scale.

```
1.  lspci -tv
```

En el resultado que muestra el comando, hay que localizar la tarjeta de red y apuntar su identificador.

Ilustración 38 Identificador tarjeta de red. Fuente: propia.

Con el identificador apuntado, hay que añadir el dispositivo en la máquina virtual y este se convierte en un dispositivo disponible en el sistema operativo a hospedar.

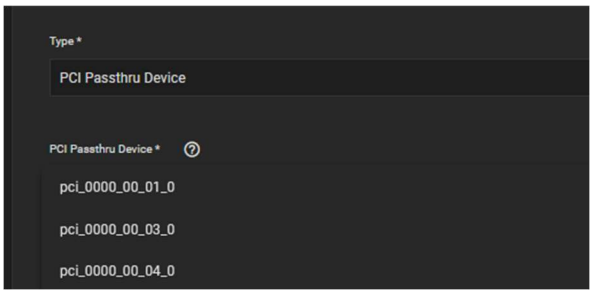

Ilustración 39 Selección dispositivo PCI-Express. Fuente: propia.

El resto de configuración de pfSense se realiza siguiendo el asistente del instalador sin problema alguno.

7.4 Configurar Dockers

Instalar o configurar contenedores Docker en TrueNAS Scale es una tarea sencilla, el principal problema reside en encontrar la opción, que está escondida bajo el nombre de Apps. Al navegar hasta esa página, en la esquina superior derecha se muestra un botón que permite añadir contenedores Docker.

El asistente de configuración solicita el repositorio de DockerHub en el que se encuentra la imagen de Docker, en el caso del servicio distribuidor de máquinas virtuales se encuentra en iskrudo/vmserver, como se aprecia en la siguiente ilustración. Es importante seleccionar “Always pull image even if present on host” para asegurarse de que siempre se trabaja con la última versión del repositorio de DockerHub.

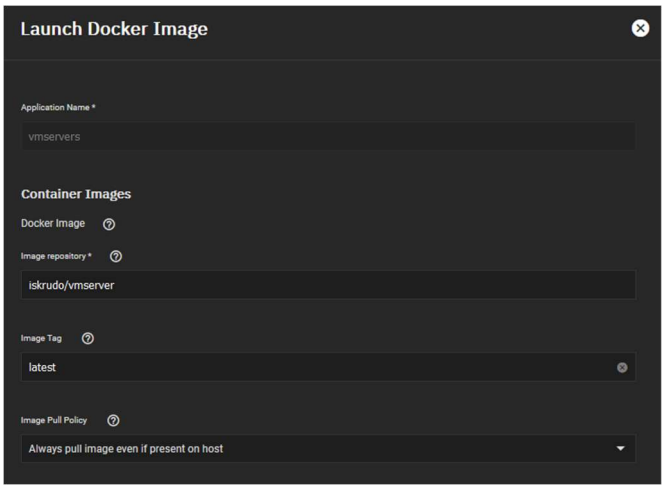

Ilustración 40 Configuración Docker Container. Fuente: propia.

Es recomendable asignar una dirección IP propia a cada contenedor Docker para evitarse problemas con los puertos. En el caso de este repositorio, se ha optado por asignar la siguiente configuración.

Es importante recalcar que al darle una dirección IP estática, se deben indicar las IP, como si de una ruta estática en un router se tratase. Una vez se ha indicado cuál es la puerta de enlace predeterminada al contenedor, se puede aceptar el dialogo y finalizar la creación.

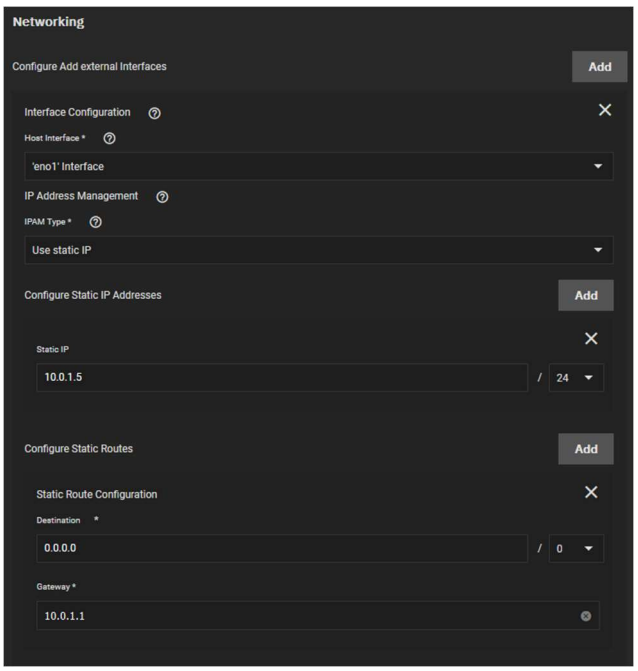

Ilustración 41 Configuración Network Docker container. Fuente: propia.

7.4.1 Creación de webAPI

Se ha escogido .Net 6 para la creación tanto de la webAPI como para el cliente. Esto es debido a la naturaleza multiplataforma de las versiones actuales de .Net. Las versiones superiores a .Net 5 (la 5 incluida) son la herencia del proyecto .Net Core, la versión de código abierto de .Net Framework que pretende funcionar en el mayor número de plataformas posible [34]. Desde la quinta entrega de la versión Core, lanzada bajo el nombre de .Net 5 (para unificar las dos ramas de la plataforma del gigante de Redmond), se han dedicado todos los esfuerzos en convertir .Net en un ecosistema abierto, multiplataforma, y con un rendimiento superior a sus versiones de legado [35].

El framework web de .Net es ASP.Net Core, un framework que funciona bajo el servidor multiplataforma Kestrel [36]. ASP.Net Core, con Kestrel, es uno de los mejores frameworks web y es considerado como el octavo con más rendimiento en la lista de TechEmpower [37].

Para crear un proyecto de webAPI en ASP.Net Core, se debe abrir una consola de comandos y escribir el siguiente comando:

```
dotnet new webapi -o VMAPI
```

Este comando crea un proyecto nuevo de ASP.Net Core con la plantilla de Web API. Esta plantilla permite al usuario crear una API de forma sencilla e incluyendo todas las funcionalidades del framework. También es posible utilizar minimal API, otro estilo de programación de APIs, pero se ha preferido usar el estilo convencional con controladores para facilitar la comprensión del código.

La web API consta de cinco clases, una de ellas generada automáticamente y modificada levemente, y una interfaz. La primera clase a tener en cuenta es Program.cs, ya que es donde se encuentra el punto de entrada de la aplicación.

```
public static void Main(string[] args)
        {
            var builder = WebApplication.CreateBuilder(args);
            builder.Services.AddControllers();

            builder.Services.AddEndpointsApiExplorer();
```

```
7.              builder.Services.AddSwaggerGen();
8.
9.              builder.Services.AddSingleton<IDatabaseService,DatabaseService>();
10.
11.             var app = builder.Build();
12.
13.             if (app.Environment.IsDevelopment())
14.             {
15.                 app.UseSwagger();
16.                 app.UseSwaggerUI();
17.             }
18.
19.             app.UseAuthorization();
20.             app.MapControllers();
21.             app.Run();
22.         }
```

El único aspecto relevante de este código es la configuración de la inyección de dependencias, que se realiza en la línea 9. Esa línea es la encargada de indicar al servicio de inyección de dependencias de .Net que se requiere de una única copia de la clase DatabaseService y que esa clase queda registrada bajo la interfaz IDatabaseService. Es posible registrar la clase directamente, sin necesidad de una interfaz, pero usar una interfaz proporciona escalabilidad y modularidad, ya que DatabaseService se puede sustituir por una implementación diferente (y más compleja) de IDatabaseService sin que el resto de las clases se vieran afectadas.

El siguiente aspecto relevante de este código es el DatabaseService.

```
1.  public class DatabaseService : IDatabaseService
2.  {
3.      private List<ConnectionModel> _availableConn;
4.      private List<ConnectionModel> _servicedConn;
5.
6.      public DatabaseService()
7.      {
8.          _availableConn = new ();
9.          _servicedConn = new ();
10.     }
11.     public ConnectionModel? GetAvailableVM()
12.     {
13.         var conn = _availableConn.FirstOrDefault();
14.         if (conn == null) return null;
15.         _availableConn.Remove(conn);
16.         _servicedConn.Add(conn);
17.         return conn;
18.     }
19.     public void ReleaseVM(ConnectionModel conn)
20.     {
21.         _availableConn.Add(conn);
22.         _servicedConn.Remove(conn);
23.     }
24.     public void HeartBeat(ConnectionModel conn)
25.     {
26.         if (!_availableConn.Exists(x => x?.Equals(conn) ?? false) &&
    !_servicedConn.Exists(x => x?.Equals(conn) ?? false))
27.         {
28.             _availableConn.Add(conn);
29.         }
```

```
30.     }
31. }
```

Este código es el encargado de la gestión de las máquinas virtuales. Se encarga de suministrar VMs (en forma de modelos de conexión), de devolverlas a la pool de conexiones, y de asegurarse de que las máquinas virtuales recién encendidas sean añadidas a la pool.

Por último, se analiza uno de los controladores, estando el resto a disposición del lector en el repositorio de Github correspondiente.

```
1.  [ApiController]
2.  [Route("[controller]")]
3.  public class HeartbeatController : ControllerBase
4.  {
5.      private IDatabaseService _database;
6.
7.      public HeartbeatController(IDatabaseService _service)
8.      {
9.          _database = _service;
10.     }
11.
12.     [HttpPost]
13.     public ActionResult Post(ConnectionModel conn)
14.     {
15.         _database.HeartBeat(conn);
16.         return Ok();
17.     }
18. }
```

Los controladores de ASP.Net Core, si son usados para una web API) deben estar marcados con el atributo [ApiController], y se debe indicar la correspondiente ruta con el atributo [Route(“”)]. En este caso, se usa “[controller]” como ruta. Esto indica a Kestrel que estos métodos están disponibles bajo la dirección /Heartbeat, ya que [controller] es sustituido por el nombre del controlador (sin controller al final).

Es importante destacar que IDatabaseService se inyecta mediante el constructor, y no mediante una anotación o un atributo.

Por último, es de vital importancia destacar que el método HTTP a usar se especifica mediante el atributo [HttpPost], [HttpGet], etc.

7.4.1.1 Creación cliente

El cliente es el encargado de solicitar una máquina virtual, establecer la conexión mediante RDP, y devolver la máquina virtual a la pool de conexiones. Esto se consigue mediante este código simple.

```
1.  string baseUrl = "http://10.0.1.5";
2.
3.  HttpClient client = new HttpClient();
4.  var res = await client.GetAsync($"{baseUrl}/Connect");
5.
6.  if (res.IsSuccessStatusCode)
7.  {
8.      var conn = await res.Content.ReadFromJsonAsync<ConnectionModel>();
9.
10.     var result = await Cli.Wrap("/usr/local/bin/xfreerdp")
11.         .WithArguments($"/v:{conn!.IpAddress}:{conn!.Port} /f /p:admin /d:
    /network:auto /rfx /multimon /rfx-mode:video")
12.         .WithValidation(CommandResultValidation.None)
13.         .ExecuteAsync();
14.
15.     await client.PostAsync($"{baseUrl}/Release",
    JsonContent.Create<ConnectionModel>(conn));
16. }
```

Los dos aspectos más relevantes del código son el uso de un HttpClient para hacer una petición de conexión al servicio mencionado anteriormente, y el CLI de Cli.Wrap para invocar un programa en Linux, que es el cliente RDP en sí. Cabe destacar que la ejecución de Cli.Wrap queda bloqueada hasta que el cliente RDP se cierra, por lo que no realiza la siguiente acción, que consiste en devolver la máquina virtual, hasta que se ha cerrado la conexión con la correspondiente máquina virtual.

7.4.1.2 Creación del worker service (heartbeat)

El Worker service es el encargado de notificar cuando una máquina virtual sigue con vida. Manda un heartbeat cada 5 minutos, notificando al servidor que sigue operativa, y si la máquina virtual es apagada, este servicio notifica al servidor de su retirada del servicio.

Esto se consigue mediante la creación de una clase que herede de BackgroundService.

Uno de los aspectos más importantes del código reside en la clase Worker, estando disponible el resto del código en su correspondiente repositorio de Github.

```
1.  public class Worker : BackgroundService
2.  {
3.      private readonly ILogger<Worker> _logger;
4.      private readonly HTTPService _httpService;
```

```
5.
6.      public Worker(ILogger<Worker> logger, HTTPService ser)
7.      {
8.          _logger = logger;
9.          _httpService = ser;
10.     }
11.
12.     protected override async Task ExecuteAsync(CancellationToken stoppingToken)
13.     {
14.         _logger.Log(LogLevel.Information, "Worker started");
15.         while (!stoppingToken.IsCancellationRequested)
16.         {
17.             _httpService.HeartBeat();
18.             _logger.Log(LogLevel.Information, "Heartbeat Sent");
19.             await Task.Delay(1000 * 60 * 5, stoppingToken);
20.         }
21.         _logger.Log(LogLevel.Information, "Worker stopped");
22.     }
23. }
```

La clase Worker es la encargada de mandar los heartbeats al servidor mediante un HTTPService. Este servicio es inyectado mediante inyección de dependencias en el constructor de la clase.

7.4.2 Creación del servidor TFTP

Para la creación del servidor TFTP se ha optado por utilizar una aplicación de consola de .Net. En esta aplicación se ha usado TFTP.Net para programar el servidor TFTP.

Uno de los métodos más importantes, estando el resto del código fuente a disposición del lector en el respectivo repositorio de Github, es ServerOnReadRequest.

```
1.  private static void ServerOnReadRequest(ITftpTransfer transfer, EndPoint client)
2.      {
3.          var path = _serverDirectory + transfer.Filename;
4.          var file = new FileInfo(path);
5.
6.          if(file.FullName == "/debian-installer/arm64/grub/grub.cfg")
7.          {
8.              file = new FileInfo("/app/images/aarch64/grub.cfg");
9.          }
10.
11.         //Is the file in the server directory?
12.         if (!file.Exists)
13.         {
14.             Console.WriteLine($"A client requested {file.FullName} but there's no file
    there.");
15.             ServerOnReadRequestRPI(transfer, client);
16.         }
17.         else
18.         {
19.             Console.WriteLine($"A client requested {file.FullName}");
20.             OutputTransferStatus(transfer, "Accepting request from " + client);
21.             StartTransfer(transfer, new FileStream(file.FullName, FileMode.Open,
    FileAccess.Read));
22.         }
23.     }
```

Este método acepta una transferencia, en la que se especifica la ruta al archivo, y el cliente al que se debe mandar el archivo. Hay un caso concreto definido en el que se debe utilizar una ruta diferente a la suministrada, pero en el resto de los casos, se usa la ruta que viene indicada en la transferencia.

Si el fichero no es encontrado, es posible que una versión antigua del firmware de Raspberry Pi se esté usando. Para compatibilizar esa versión del software, se ha decidido crear un segundo método, ServerOnReadRequestRPI, que se encarga de gestionar ese caso concreto.

7.5 Preparar Raspberry Pi

Desde la revisión B de la placa Raspberry Pi 4, todos los firmwares instalados son compatibles con el arranque en red. En caso de que se disponga de una versión antigua del firmware, o de una Raspberry Pi 3, se debe flashear el nuevo firmware. Para ello se usa el programa Raspberry Pi Imager.

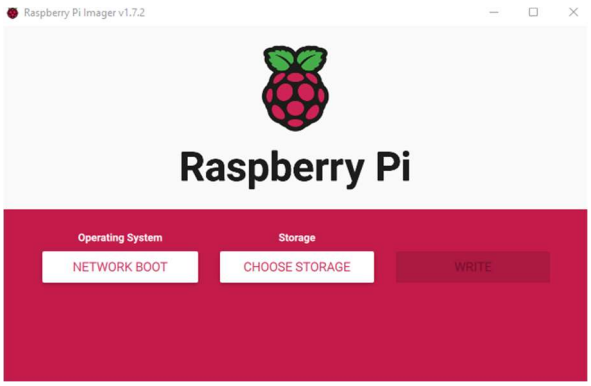

Ilustración 42 Raspberry Pi Imager. Fuente: propia.

En la pestaña de selección de sistema operativo se navega hasta Misc. Utilities, se hace clic en Bootloader, y se selecciona Network Boot. Después se debe seleccionar un dispositivo de almacenamiento desde el que se va a ejecutar la herramienta para flashear el nuevo firmware. Esta unidad de almacenamiento debe ser una tarjeta microSD.

Cuando la utilidad haya finalizado, ya se puede introducir la tarjeta microSD en la Raspberry. El parpadeo de forma reglar del led verde indica que el proceso de flashing ha finalizado.

7.6 Copiar el sistema de ficheros de una imagen

Para poder arrancar el sistema, se debe descargar una imagen de Raspberry Pi OS y copiarla en la raíz de la Pool1, un nivel antes de las carpetas de los shares. Una vez se haya copiado satisfactoriamente, se debe montar la imagen de disco y se procede a realizar una copia del sistema de ficheros de la imagen al share NFS con el siguiente comando.

```
1. rsync -xa --progress /mnt/img/ /mnt/Pool1/NFS
```

Una vez haya finalizado el comando anterior, se debe copiar la carpeta boot a su share correspondiente con el siguiente comando.

```
1. cp -r /mnt/img/boot/* /mnt/Pool1/images/
```

7.7 fstab

Para que Linux sepa encontrar correctamente las carpetas /boot y /tmp hay que configurar el archivo /etc/fstab. El fichero debe quedar con la siguiente configuración, permitiendo así que /tmp sea un disco en RAM y /boot esté localizado en el otro share.

```
1. proc            /proc           proc    defaults          0       0
2. 10.0.1.3:/mnt/Pool1/images      /boot nfs defaults,vers=3 0       0
3. tmpfs /tmp tmpfs defaults,noatime,nosuid,nodev,noexec,mode=1777,size=256M 0 0
```

7.8 FreeRDP

No existe una versión compilada de FreeRDP para la versión de 64bits de Raspberry Pi OS. Por ende, se debe recompilar el código del repositorio de Github.

Para ello, se debe clonar el repositorio de Github en una carpeta, y ejecutar make, que se encarga de crear los binarios necesarios para la ejecución de FreeRDP. Una vez finalice el comando, se debe ejecutar make Install, que se encarga de instalar la aplicación correctamente en el directorio /usr/local/bin.

7.9 Resultado Final

Con todos los componentes listos para la ejecución de la secuencia descrita a continuación, solo se requiere de la creación y registro de un servicio en systemd que se encargue de ejecutar el cliente de VM Service.

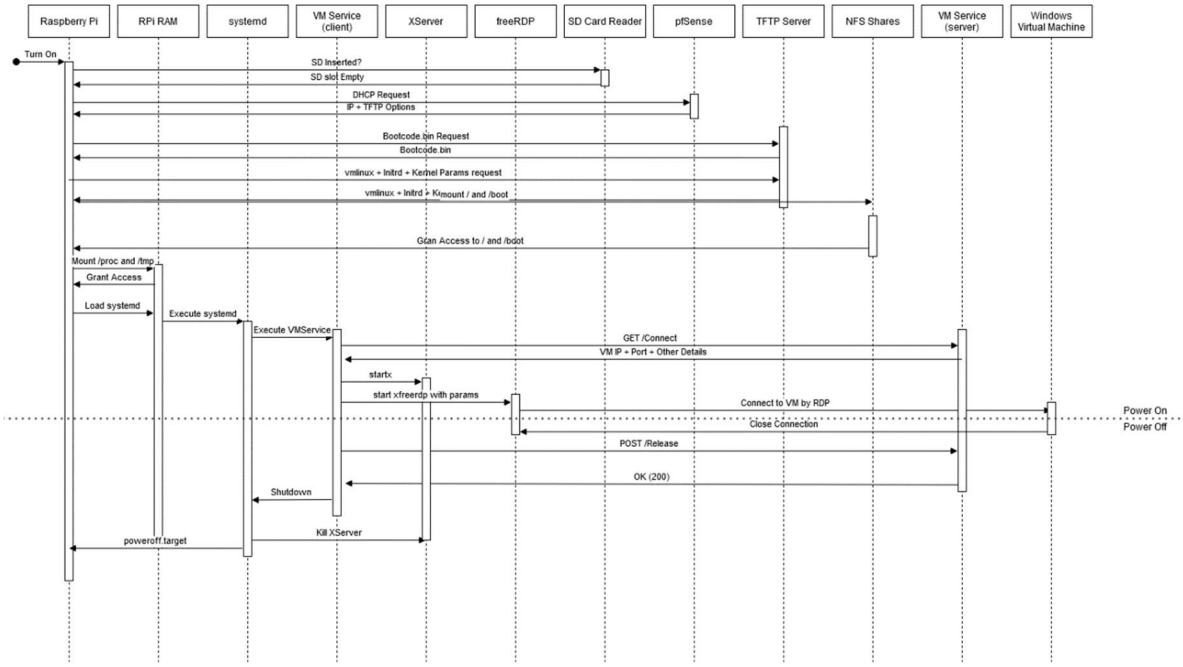

Ilustración 43 Diagrama secuencia boot Raspberry Pi. Fuente: propia.

Para ello, se debe crear un fichero en /lib/systemd/system con el nombre deseado para el servicio. El archivo debe tener el siguiente contenido.

```
[Unit]
Description=RDP Service
After=multi-user.target
[Service]
User=admin
Type=unique
ExecStart=/usr/local/bin/rdpclient/rdp.sh
[Install]
WantedBy=multi-user.target
```

Todo este desarrollo y configuración permiten que, al conectar un dispositivo Raspberry Pi sin ningún tipo de soporte de almacenamiento a la red, el dispositivo se convierta en un cliente ligero y se conecte de forma autónoma a una máquina virtual en tan solo 55 segundos.

Capítulo VIII: Ampliaciones

Las posibles ampliaciones de este proyecto son abundantes y diversas en cuanto a la adición de nuevas funcionalidades; si lo que se busca obtener es una mejora de los sistemas actuales, se han concebido dos posibles mejoras a la infraestructura actual.

8.1 UEFI y bootstrapping

Esta primera sección del capítulo de ampliaciones trata sobre la migración a un firmware UEFI de las Raspberry Pi objetivo, sus consecuencias, y el uso de bootstrapping para desplegar una solución similar a la ofrecida.

El firmware de la Raspberry Pi es un firmware propietario que no sigue el estándar UEFI. Esto hace que se deba compilar (y reprogramar ciertas secciones del arranque) de cada sistema operativo que se pretenda utilizar en la placa. Sin embargo, esto no tiene por qué ser así, ya que la CPU de Broadcom cumple con el set de instrucciones ARMv8.

La mejor solución sería que el propio fabricante decidiera hacer el firmware de Raspberry Pi compatible con UEFI, pero ante la negativa de este, la comunidad de desarrolladores ha conseguido portar TianoCore EDK2 a Raspberry Pi.

TianoCore [38] es una implementación de código abierto del estándar UEFI desarrollado por Unified EFI Forum, antes desarrollado por Intel.

Al portar esta implementación, se convierte a la Raspberry Pi en un dispositivo compatible con ACPI y UEFI.

ACPI [39] es una interfaz común definida por la industria que se encarga de listar y administrar el hardware, y gestionar sus estados energéticos. Esta interfaz viene definida

como una tabla en la que se listan los dispositivos y sus características. Esto es especialmente útil para la compatibilidad entre placas base y sistemas operativos, ya que la propia placa base dota al sistema operativo de toda la información necesaria para controlar el dispositivo. Mediante el uso de esta tabla, es posible que diferentes dispositivos que cumplan con ACPI y usen el mismo set de instrucciones funcionen con un mismo compilado de un sistema operativo, sin importar la marca o las características de cada dispositivo.

Se ha probado a cargar el firmware mediante TFTP con resultados excepcionales, ya que se ha conseguido arrancar el firmware UEFI, que, a su vez, ha descargado del servidor TFTP un ejecutable de GRUB compilado para ARM64 y lo ha ejecutado sin problema alguno. Con el cargador de arranque cargado, debe ser posible cargar cualquier distribución Linux compatible con ARM64 siguiendo los pasos mostrados en la siguiente ilustración.

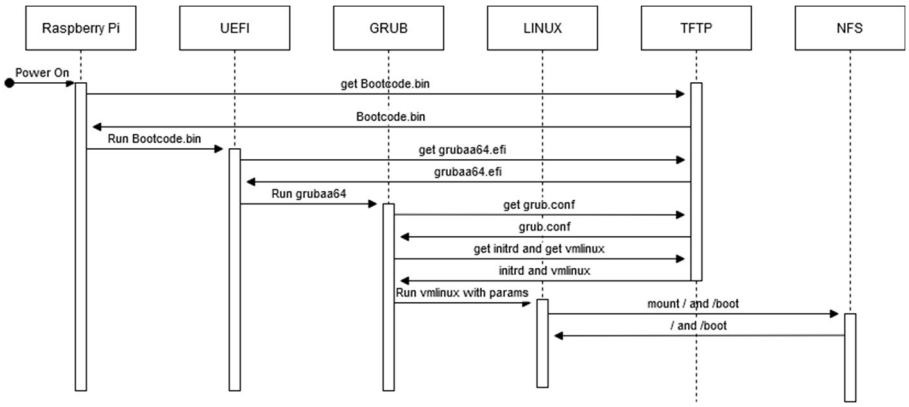

Ilustración 44 Diagrama de secuencia bootstrapping UEFI. Fuente: propia.

Sin embargo, a pesar de que GRUB carga perfectamente, el inicio de la imagen Linux se ve frustrado por un error de sincronismo, como se muestra a continuación.

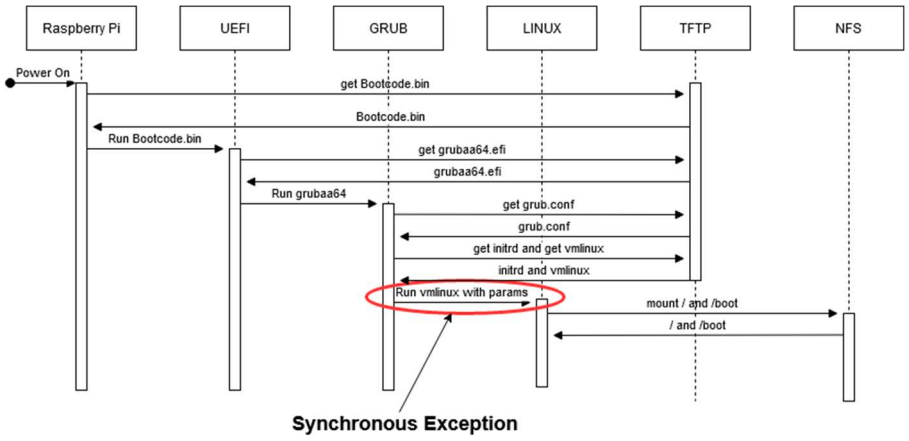

Ilustración 45 Diagrama de secuencia bootstrapping con error. Fuente: propia.

Se desconoce la magnitud del error, aunque la documentación de ARMv8 propone una serie de posibles causantes del error [38]:

- La MMU se encuentra con algún error que la hace abortar el proceso.
- Falla la comprobación de alineación de SP y PC.
- Fallos externos sincrónicos.
- Uso de instrucciones no asignadas.
- Lanzamiento de excepciones por parte del código.

En cualquier caso, con más cantidad de tiempo y una investigación exhaustiva de la documentación de ARM, es factible este método de arranque que aumenta la compatibilidad del sistema, favorece la interoperabilidad, y encamina el proyecto hacia los estándares de la industria, permitiendo a otros dispositivos ARM, compatibles con UEFI, volverse compatibles con el sistema.

8.2 Active Directory

Esta segunda sección trata sobre la inclusión de Active Directory como método para autenticar a los usuarios.

Esto se conseguiría con una pantalla de inicio de sesión en la Raspberry Pi, que transmitiría las credenciales mediante HTTPS al servidor. Este servidor debería comprobar las credenciales mediante el uso de System.DirectoryServices.AccountManagement, un paquete de .Net que permite hacer consultas a Active Directory. Con los datos verificados, el servidor respondería con la máquina virtual tal y como funciona en la actualidad.

Capítulo IX: Conclusiones

En este trabajo de fin de grado se ha estudiado la viabilidad de utilizar Raspberry Pis como clientes ligeros, y se ha puesto en marcha el prototipo, totalmente funcional, de una solución basada en este concepto.

En el marco teórico se ha estudiado las diferentes tecnologías que intervienen en la conexión remota a dispositivos y la virtualización, se ha repasado brevemente la historia de la comunicación y los clientes ligeros, se han analizado los diferentes protocolos que intervienen en el arranque en red, se ha estudiado la secuencia de arranque de una Raspberry Pi, se ha investigado acerca del arranque usando BIOS y UEFI, se ha analizado la secuencia de arranque de Linux tanto en x86_64 como en ARM, y se han observado las principales diferencias entre SysVinit y systemd.

En la etapa de desarrollo se ha conseguido cumplir con todos los objetivos primarios, y por ende todos los requisitos primarios, propuestos al principio del proyecto. Los requisitos funcionales secundarios han sido completados casi en su totalidad, solo faltando el requisito de transmitir dispositivos vía USB/IP, y los requisitos no-funcionales tanto primarios como secundarios se han cumplido con éxito.

Gran parte del éxito de este proyecto reside en la utilización de tecnologías ampliamente conocidas, como puede ser .Net, y en la dedicación de un tiempo significativo en la investigación y búsqueda de fuentes tanto fiables como no-fiables. Con estas últimas, se ha tenido que realizar un trabajo de prueba y error para determinar qué fuentes podían ser consideradas y qué fuentes se encuentran desactualizadas, tienen instrucciones erróneas, o muestran soluciones que no se adaptan a las necesidades del proyecto.

La metodología escogida para este proyecto ha demostrado ser eficaz, pese a que la planificación inicial ha debido ser modificada para aumentar el número de horas de investigación y disminuir, en igual medida, el número de horas de desarrollo.

En cuanto al Testing, se ha optado por testear los requisitos funcionales y no funcionales de forma manual, comprobando si el sistema los cumple o no.

En resumen, este proyecto ha demostrado que esta solución puede ser implementada en pequeñas o medianas empresas y constituye un ahorro significativo a nivel energético, proporciona un menor mantenimiento y lo concentra en un dispositivo clave (como es el servidor de virtualización), y contribuye a mantener la infraestructura lista para ser usada el mayor tiempo posible (minimizando así el downtime).

9.1 Catálogo de conocimiento

Para poder realizar este trabajo se ha requerido de los siguientes conocimientos:

- Conocimientos en Docker.
- Conocimientos de virtualización con KVM.
- Conocimiento en VirtIO.
- Desarrollo en .Net.
    - Desarrollo en ASP.NET Core.
    - Desarrollo de aplicaciones de consola.
    - Desarrollo de Servicios en segundo plano.
- Conocimientos del Arranque de Linux.
    - Conocimientos de GRUB.
    - Conocimientos de systemd.
- Conocimientos en UEFI y BIOS.
- Conocimientos en administración de servidores Dell.
- Nociones básicas de redes.
- Conocimientos en pfSense.

Capítulo X: Bibliografía

[1] Z. Bauman, Modernidad líquida, Fondo de Cultura Economica, 2004.

[2] D. Fresneda, «Radiografía de la temporalidad en España,» RTVE.es, 23 12 2021. [En línea]. Available: https://www.rtve.es/noticias/20211223/temporalidad-empleo-espana/2103964.shtml. [Último acceso: 28 12 2021].

[3] T. Aburto, E. Martín y A. Hernández, «El gas, el 'arma' de Putin para maniatar a Europa en la crisis de Ucrania,» El Mundo, 03 02 2022. [En línea]. Available: https://www.elmundo.es/internacional/2022/02/03/61f9a1a3e4d4d8805a8b45e2.html. [Último acceso: 04 02 2022].

[4] Information Processing Society of Japan, «IPSJ Computer Museum,» Information Processing Society of Japan, [En línea]. Available: http://museum.ipsj.or.jp/en/computer/personal/0086.html. [Último acceso: 08 01 2022].

[5] «Mouser,» Mouser, [En línea]. Available: https://www.mouser.es/ProductDetail/Microchip-Technology-Atmel/ATMEGA328P-AU?qs=K8BHR703ZXiCmmgp6%2FGNmQ%3D%3D. [Último acceso: 30 01 2022].

[6] N. Negroponte, El mundo digital, Barcelona: Ediciones B, S.A., 1995.

[7] A. Tugui, *Calm technologies in a multimedia world,* 2004.

[8] E. Arrieta, «Cultura Genial - El hombre es un ser social por naturaleza,» Cultura Genial, [En línea]. Available: https://www.culturagenial.com/es/el-hombre-es-un-ser-social-por-naturaleza/. [Último acceso: 13 01 2022].

[9] M. Bunson, Encyclopedia of the Roman Empire (Facts on File Library of World History), Facts on File, 2002.

[10] F. Telefónica, Telégrafos. Un relato de su travesía, Barcelona: Fundación Telefónica, 2014.

[11] J. Yang, J. Nieh, M. Selsky y N. Tiwari, «The Performance of Remote Display Mechanisms for Thin-Client Computing,» de *USENIX 2002*, Monterey, 202.

[12] M. Schofield, «Documentación técnica de Microsoft,» Microsoft, [En línea]. Available: https://docs.microsoft.com/es-es/windows/win32/termserv/remote-desktop-protocol?redirectedfrom=MSDN. [Último acceso: 1 12 2021].

[13] G. F. Pinzari, «NX X Protocol Compression,» 2003.

[14] S. M. Jain, Linux Containers and Virtualization: A Kernel Perspective, Bengaluru: Springer Science+Business Media LLC, 2020.

[15] A. Silberschatz, Operating System Concepts, Wiley, 2009.

[16] T. Deshane, Z. Shepherd, J. N. Matthews, M. Ben-Yehuda, A. Shah y B. Rao, «Quantitative Comparison of Xen and KVM,» Xen Summit, Boston, 2008.

[17] A. Chierici y R. Veraldi, «A quantitative comparison between xen and kvm,» de *17th International Conference on Computing in High Energy and Nuclear Physics (CHEP09)*, 2010.

[18] C. D. Graziano, «A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project,» Iowa State University, Iowa, 2011.

[19] J. DORDOIGNE, Redes informáticas - Nociones fundamentales (5ª edición), Cornellà de LLobregat: Ediciones ENI, 20015.

[20] Novell, Novell Netware 3.11, Novell, 1991.

[21] Intel Corporation; SystemSoft, «Preboot Execution Environment (PXE) Specification,» Intel Corportaion, 1999.

[22] E. Ariganello, Redes Cisco, Paracuellos de Jarama: Ra-Ma Editorial, 2020.

[23] IBM, «IBM Documentation,» IBM, [En línea]. Available: https://www.ibm.com/docs/en/aix/7.2?topic=management-network-file-system. [Último acceso: 07 06 2022].

[24] Raspberry Pi Foundation, «Raspberry Pi Github Documentation,» 08 03 2022. [En línea]. Available: https://github.com/raspberrypi/documentation/blob/develop/documentation/asciidoc/computers/raspberry-pi/bootflow-2711.adoc. [Último acceso: 24 03 2022].

[25] G. Duarte, «Many But Finite,» 05 06 2008. [En línea]. Available: https://manybutfinite.com/post/how-computers-boot-up/. [Último acceso: 24 03 2022].

[26] A. S. Tanenbaum, Modern Operating Systems, 3rd ed., PHI Learning, 2007.

[27] W. Stallings, Operating systems: Internals and Design Principles, 7th ed, Prentice Hall, 2012.

[28] UEFI, «Unified Extensible Firmware Interface (UEFI) Version 2.9,» 2021.

[29] D. P. Bovet y M. Cesati, Understanding the Linux Kernel, Third Edition, Sebastopol: O'Reilly, 2005.

[30] M. Russinovich y D. A. Solomon, Windows® Internals: Including Windows Server 2008 and Windows Vista, Fifth Edition, Redmond: Microsoft Press, 2009.

[31] Netgate, «pfSense,» [En línea]. Available: pfSense.org. [Último acceso: 20 04 2022].

[32] Dell Technologies, «Cómo configurar la dirección IP de iDRAC mediante el panel LCD,» Dell, [En línea]. Available: https://www.dell.com/support/kbdoc/es-es/000124010/c%C3%B3mo-configurar-la-direcci%C3%B3n-ip-de-idrac-mediante-el-panel-lcd. [Último acceso: 06 06 2022].

[33] I. Patterson, «NSSM - the Non-Sucking Service Manager,» Iain Patterson, [En línea]. Available: https://nssm.cc/. [Último acceso: 14 06 2022].

[34] I. Landwerth, «.NET Core is Open Source - .Net Blog,» Microsoft, 12 11 2014. [En línea]. Available: https://devblogs.microsoft.com/dotnet/net-core-is-open-source/. [Último acceso: 04 06 2022].

[35] S. Toub, «Performance Improvements in .NET 5 - .Net Blog,» Microsoft, 13 07 2020. [En línea]. Available: https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-5/. [Último acceso: 10 06 2022].

[36] T. Dykstra, C. Ross y S. Halter, «Kestrel web server implementation in ASP.NET Core - Microsoft Docs,» Microsoft, 06 03 2022. [En línea]. Available: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-6.0. [Último acceso: 10 06 2022].

[37] TechEmpower, «Web Frameworks Benchmark - TechEmpower,» TechEmpower, 08 02 2021. [En línea]. Available: https://www.techempower.com/benchmarks/#section=data-r20&hw=ph&test=composite. [Último acceso: 10 06 2022].

[38] TianoCore, «What is TianoCore? - TianoCore,» TianoCore, [En línea]. Available: https://www.tianocore.org/. [Último acceso: 13 06 2022].

[39] UEFI Forum, «Advanced Configuration and Power Interface (ACPI) Specification,» UEFI Forum, 2020.

[40] ARM, «ARM Cortex-A Series Programmer's Guide for ARMv8-A,» ARM, [En línea]. Available: https://developer.arm.com/documentation/den0024/a/AArch64-Exception-Handling/Synchronous-and-asynchronous-exceptions. [Último acceso: 22 05 2022].

[41] iXSystems, «TrueNAS,» [En línea]. Available: https://www.truenas.com. [Último acceso: 06 06 2022].