TREBALL FINAL DE GRAU

TÍTULO DEL TFG: Automatización y despliegue de servidores en un clúster privado

TITULACIÓN: Grado en Ingeniería Telemática

AUTOR: Manuel Alejandro Baena Navarro

DIRECTOR: Toni Oller Arcas

SUPERVISOR: Gerard Solé Castellví

Títol: Automatización y despliegue de servidores en un clúster privado

Autor: Manuel Alejandro Baena Navarro

Director: Toni Oller Arcas

Supervisor: Gerard Solé Castellví

Data: 28 de juny del 2016

Resum

En este proyecto se intenta dar solución a la problemática de una empresa haciendo uso de un hardware disponible y sin uso de esta.

Se intentará dar solución a la gestión ineficiente de los recursos tecnológicos de su centro de datos y la vez proporcionar herramientas para la gestión y administración de estos recursos de una manera centralizada.

En este proyecto se presentan dos fases, por un lado tenemos toda una plataforma de arranque en la que se despliega todo un clúster operativo de manera automatizada, todas las máquinas obtienen tanto su configuración, sistema de almacenamiento y sistema operativo vía red. Al final del proceso todas las máquinas pueden realizar operaciones de virtualización de manera efectiva. Esta fase del proyecto ha sido la mas larga y complicada.

La segunda fase consiste en el despliegue de una plataforma de gestión capaz de gestionar los recursos del clúster, centralizar la administración, proporcionar un sistema de monitorización y facilitar tareas como las migraciones de máquinas virtuales dentro de las máquinas del clúster.

Si bien el proyecto consta de muchas partes complejas, solamente se ha requerido de un esfuerzo de configuración e integración de todas las partes.

Title: Automatización y despliegue de servidores en un clúster privado

Author: Manuel Alejandro Baena Navarro

Director: Toni Oller Arcas

Date: June 28 rd 2016

Overview

This project attempts to solve the problems of a company using a available and unused hardware.

It will try to solve the inefficient management of technology resources of the data center while providing tools for management and administration of these resources in a centralized manner.

In this project two phases are presented, on the one hand we have a whole boot platform in which an entire operating cluster is deployed in an automated manner, all machines get both its configuration, storage system and operating system via network. At the end of the process all machines can operate effectively virtualization. This phase of the project has been the longest and most complicated.

The second phase consists of the deployment of a management platform capable of managing cluster resources, centralize management, provide a monitoring system and facilitate tasks such as migration of virtual machines within the cluster computers.

While the project consists of many complex parts, but it has required an effort of configuration and integration of all parties.

Un agradecimiento a mi madre, pues sin su esfuerzo y sacrificio no podría estar haciendo este trabajo.
A Júlia por animarme a estudiar cuando pensaba que no valía para ello.
A Gerard Solé por compartir conmigo lo que sabe y ayudarme con el proyecto.
A Roxana, Andrea y Sandra por ser fuente de ánimo y apoyo durante todo este tiempo.

ÍNDICE

ÍNDICE DE CUADROS

INTRODUCCIÓN

El fuerte crecimiento de internet, los servicios web, aplicaciones han proliferada lo aparición y el crecimiento de los centros de datos a lo largo de todo el mundo. Estos centros de datos están expuestos internet prácticamente en toda su totalidad, pero lo que exponen a internet es una pequeña parte de una infraestructura mas compleja con servidores internos no expuestos a internet y realizando diversas funciones dentro del centro de datos.

Estos centros de datos por lo general son grandes infraestructuras compuestas por centenares o miles de servidores, que proporcionan múltiples servicios pero que a la vez son gestionados por grupos de personas reducidos. Estos grupos de personas tienen que lidiar diariamente con problemas como la gestión de recursos, soluciones de fallos, reemplazo de recursos que quedan inoperativos y problemas de escalabilidad.

Las empresas grandes con centros de datos de volúmenes enormes generalmente disponen de recursos y medios para desarrollar tecnologías que permitan la gestión de sus centros de datos. El resto de empresas tienen que depender de la innovación tecnológica de estas y del esfuerzo de la comunidad para desarrollar tecnologías que desafían esta problemática. La complejidad y los problemas de los centros de datos a gran escala es extrapolable en su medida a centros de datos medianos y pequeños.

Los dos desafíos principales que se encuentran los equipos de gestión de centros de datos son:

Uso eficiente de los recursos, según un estudio realizado por Gartner [1] las empresas empleaban tan solo un 12% de los recursos disponibles en sus centros de datos identificando como principales causas:

- La asignación de recursos de manera estática
- La complejidad del software actual ejecutado, y las estructuras implementadas con este, dificulta la capacidad para prever los recursos físicos necesarios derivando así en una asignación incorrecta de recursos.

Aumento de la complejidad de administración, la complejidad de administración de un centro de datos depende directamente de la cantidad de servidores y servicios a gestionar. Pero esta complejidad se dispara si estos servidores y servicios son diferentes. Es decir es mucho más fácil gestionar servidores iguales que gestionar un número inferior de servidores pero diferentes.

En este proyecto se intentará dar solución a un problema de una empresa, Alteraid S.L spinoff de UPC que tiene un convenio con un grupo de investigación de UPC para el desarrollo de proyectos, con la gestión y uso de los recursos en su centro de datos, creando una plataforma que permita el uso más eficiente de los recursos de sus servidores, facilite la gestión de estos por parte del equipo técnico y permita obtener trazas e información del estado de las máquinas en todo momento.

Para ello se utilizará un conjunto homogéneo de servidores gestionados por una plataforma de gestión que se encargará de controlar los diferentes servicios, facilitará la gestión de los recursos de manera más eficiente y proporcionara herramientas para controlar el estados de los servidores de manera clara y centralizada.

En el capitulo 1 de este proyecto se analizará la forma de trabajo y la problemática de la empresa para poder realizar un análisis de necesidades. Una vez hecho esto se procederá al diseño de una solución que intentará satisfacerlas.

En el capitulo 2 se analizará el hardware, el almacenamiento disponibles y se confeccionará una esquema de red sobre el que trabajaremos.

En los capítulos 3 y 4 se detallarán las dos fases del proyecto, la primera hará hincapié en el despliegue automatizado de los servidores y la segunda en el entorno de gestión.

Finalmente en el capítulo 5 se hará una pequeña guía de configuración para la implementación de el proyecto.

CAPÍTULO 1. SITUACIÓN PREVIA Y PROPUESTA DE SOLUCIÓN

En este capítulo se detallará la problemática que tiene la empresa en referencia a su sistema de trabajo en la nube y virtualización. Se propondrá una solución para esta problemática así como se analizaran diversas opciones disponibles en el mercado para aplicarla.

1.1 Situación previa

La principal problemática que tiene la empresa en cuestiones de trabajo en la nube y virtualización es a nivel de gestión tanto de los recursos como de las máquinas virtuales.

El método de trabajo empleado hasta la fecha consistía en desplegar manualmente máquinas virtuales, es decir:

- Comprobar máquina a máquina la carga de trabajo que tiene cada una
- Crear la máquina virtual en la máquina escogida.
- Asignar direccionamiento y red de manera manual, consultando que IP queda disponible
- Asignar los servicios en la maquina virtual

Además de la poca eficiencia en este método de trabajo también podemos encontrar problemas más puntuales pero que requieren de solución pues pueden llegar a ser críticos.

- Capacidad de migrar una máquina en caso de exceso de trabajo
- Capacidad de desplegar diferentes instancias de la misma máquina virtual en momentos de picos de trabajo

Aparte de resolver esta casuística, también se dar desea un paso más allá a nivel de gestión, como por ejemplo permitir que los empleados puedan desplegar máquinas virtuales en el centro de datos de forma fácil y controlada para poder realizar pruebas o no sobrecargar sus estaciones de trabajo. O realizar una revisión de manera rápida y fácil de como se están empleando los recursos del clúster.

Por ello la solución debe aportar:

- Un sistema de usuarios y perfiles para controlar y acotar la cantidad de recursos que puede utilizar cada empleado.

- Un sistema de monitorización constante que ofrezca información a tiempo real de el estado del clúster y la utilización de sus recursos
- Una interfaz gráfica de manera que facilite el despliegue de máquinas virtuales a los empleados que no son del área de sistemas en el caso que sea necesario.

Por lo tanto podemos definir los **requisitos** a satisfacer:

- Un sistema que posibilite la **gestión eficiente** de los recursos tecnológicos del clúster
- Facilitar el trabajo de los administradores a la hora de **gestionar** los **recursos virtuales**, como crear maquinas virtuales o migrarlas según las condiciones de uso
- Capacidad para la creación de **perfiles** de usuario
- **Monitorización** de clúster y ofrecer información de manera clara y centralizada
- Una **interfaz web** para facilitar tareas de administración

1.2 Solución propuesta

Para no parar las máquinas que ya están proporcionando servicios a la empresa y no parar ni entorpecer el funcionamiento de esta. El proyecto se realizará sobre un hardware que la empresa dispone y que hasta el momento no se estaba utilizando.

Por lo tanto la solución se ha pensado teniendo en cuenta el hardware disponible y teniendo en mente que el hardware que actualmente está en producción sea capaz de “adherirse” a ella en el futuro en caso de que los resultados cumplan la expectativas.

El hardware disponible consiste en:

- 8 máquinas con capacidad de virtualización, capacidad de conexión a red y sin capacidad de almacenaje
- 1 máquina con conexión por red, y que dispone de una controladora de discos con una capacidad de albergar hasta 16 discos.

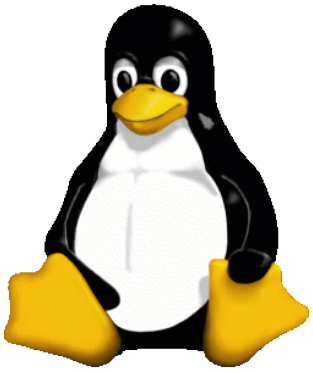
Teniendo en cuenta el hardware disponible se ha propuesto como solución la realización de un clúster de computación en el cual:

- Se empleen las primeras 8 máquinas como “músculo” del clúster, es decir, serán las máquinas destinadas a la ejecución de las máquinas virtuales y los servicios que se desplieguen en esta.

- La máquina que dispone de la controladora de discos se dedicará a orquestar todo el funcionamiento del clúster. Desde la asignación de las máquinas virtuales y su gestión, el control de los recursos y la monitorización del clúster

1.3 Tecnologías empleadas

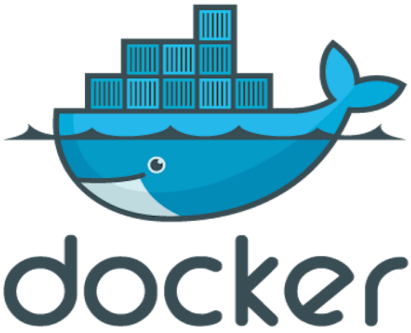
El diseño y la implementación de plataforma propuesta en este proyecto se cimienta sobre diferentes tecnologías que interactúan entre sí, de manera que en conjunto ofrecen una plataforma capaz de gestionar datos y recursos. Dentro de estas tecnologías las más destacables son:



GNU/Linux [2] es el términos empleado para referirse a la combinación del núcleo o *kernel* libre similar a Unix denominado Linux con el sistema operativo GNU. Todo su código fuente puede ser utilizado, modificado y redistribuido libremente. Es el sistema operativo más común en servidores y supercomputadoras de todos los centros de datos del mundo. En este proyecto se ha empleado Ubuntu Server como distribución Linux empleada.

KVM o **Kernel-based Virtual Machine [**3] es una solución que permite implementar virtualización de manera completa sobre Linux. Está formado por un módulo del kernel de Linux y un conjunto de herramientas de usuario. Este módulo está disponible en el kernel de Linux desde la versión 2.6.20

Xen [4] es otra solución de virtualización sobre Linux, a diferencia de KVM Xen no es un módulo del kernel sino que es un microkernel en sí. Además de la virtualización completa que ofrece KVM, Xen también es capaz de implementar paravirtualización.



Docker [5] es un proyecto de código abierto capaz de automatizar el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización en el nivel de virtualización de sistema operativo sobre Linux.[1] Docker hace uso de las utilidades de aislamiento de recursos del núcleo de Linux. Si bien no se ha

empleado Docker para el despliegue de la infraestructura propuesta, es una pieza importante en la forma de trabajo de la empresa. Pues es la solución preferida para desplegar los servicios dentro de las máquinas virtuales.

1.3.1 Orquestador

Si, el hecho de escoger las tecnologías mencionadas en el apartado anterior no ha supuesto una dificultad alguna. A la hora de escoger el orquestador no ha sido tan fácil pues existen diversas soluciones en el mercado, como es imposible analizar todas en el tiempo necesario para el desarrollo de este proyecto. Se ha estudiado dos posibles soluciones bastante populares, OpenStack y OpenNebula.

OpenStack [6] es un proyecto de computación en la nube que permite proporcionar soluciones de Infraestructura como servicio (IaaS)
OpenStack es un proyecto de software libre y código abierto bajo la licencia apache y desarrollado por la fundación OpenStack. Formada por empresas como RedHat, Canonical, Cisco entre otras 200.

OpenStack tiene una arquitectura modular en la que cada componente desempeña una función:

Módulo de computación “Nova”: Es el módulo principal, se encarga de gestionar los servicios de IaaS así como de la gestión de recursos de la plataforma y sus máquinas virtuales

Módulos de almacenamiento “swift” y “cinder”: módulos que gestionan el almacenamiento en el clúster.

Módulo de red “Neutrón”: es el encargado de gestionar las redes y las direcciones IP

Ha fecha de inicio de este proyecto cada módulo de OpenStack tenía que estar desplegado en una máquina diferente. En versiones más recientes, los módulos pueden estar desplegados en la misma máquina.

OpenNebula [7] es una plataforma para computación en la nube orientado a centros de datos distribuidos y heterogéneos, proporcionando la infraestructura virtual para

construir nubes privadas, públicas, e implementaciones híbridas de infraestructura como servicio (IaaS). OpenNebula es software libre y de código abierto bajo la licencia Apache 2

OpenNebula incluye herramientas características para la integración, administración, escalabilidad, seguridad y contabilidad, así como mecanismos para la estandarización, la interoperabilidad y la portabilidad, proporcionando a los usuarios y administradores de la nube la posibilidad de escoger entre varias interfaces de nube (Amazon EC2 Query, OGF Open Cloud Computing Interface y vCloud) así como entre diferentes hipervisores (Xen, KVM y VMWare), para acomodar múltiples combinaciones de software y hardware en un único centro de datos.

A nivel de gestión ambas tecnologías ofrecen una solución que satisface todos los requisitos mencionados con anterioridad:

- Facilitan y centralizan el despliegue de recursos virtuales en la infraestructura de la empresa
- Permite un mejor control de los recursos y facilita su administración
- Disponen de sistemas de usuarios y cuotas
- Proporciona un sistema de monitorización de los recursos
- Disponen de sistemas con interfaz gráfica para una mayor facilidad de uso y visualización de la información.

Sin embargo hay un punto clave que ha decantado la balanza en favor de OpenNebula y es el hecho de que la estructura modular de OpenStack cuya restricción de que los 4 módulos principales tienen que desplegarse en diferentes máquinas obligarían a destinar 3 servidores mas a la plataforma de gestión perdiendo recursos que podrían destinarse al despliegue de servicios.

En la siguiente se puede apreciar la relación entre los distintos componentes:

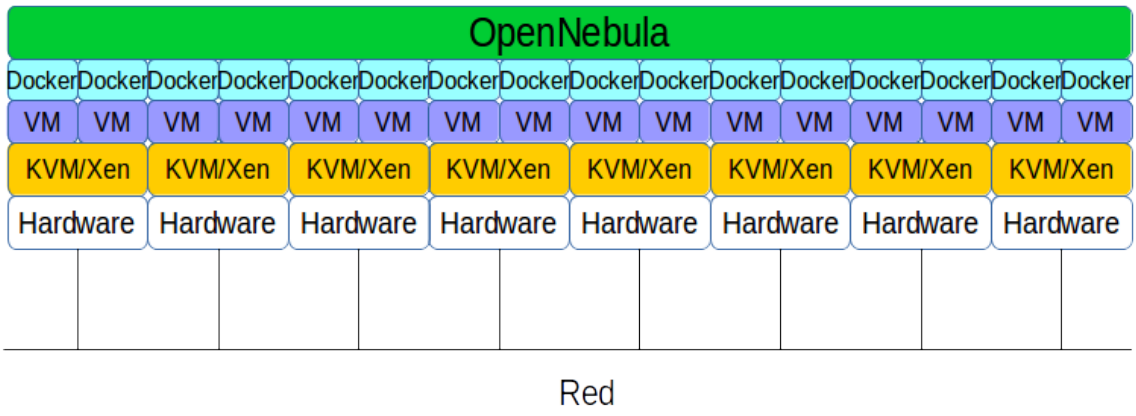


Fig 1.1 Arquitectura de la plataforma

Como se puede observar el grueso del trabajo no se centra en la explotación de la plataforma y sus recursos sino la creación de la misma y su automatización. A lo largo de los siguientes capítulos se hablará con más detalle:

- El proceso de arranque para disponer de un clúster funcional de manera automatizada, empleando distintos servicios como servidores NFS, DHCP y TFTP entre otros.
- El funcionamiento y la infraestructura de la plataforma de gestión.
- El despliegue y configuración tanto del arranque como de la plataforma de gestión.

CAPÍTULO 2: CONJUNTO DE EXPLOTACIÓN

En este capítulo se detalla la plataforma a nivel de hardware y red donde se despliega toda la infraestructura del clúster. Veremos el hardware empleado y como se ha diseñado la red para esta. El diseño de la plataforma se ha ideado teniendo en cuenta lo máximo posible los requisitos indicados en la introducción en función del hardware disponible.

2.1 Hardware

Para el proyecto se han utilizado 8 equipos que funcionarán como hipervisores del clúster, estos equipos carecen de sistema de almacenamiento físico por lo que se deriva toda la persistencia de datos a un último equipo que proporciona tanto el sistema operativo como el sistema de ficheros. Todos los equipos empleados en la plataforma son rack "*awareness*" por lo que pueden ser dispuestos en un rack para facilitar su alojamiento y administración.

La elección del hardware no responde a ninguna necesidad de la plataforma sino que la plataforma se ha diseñado en función del hardware disponible, en este caso se disponía de 8 equipos operativos con las siguientes características:

- CPU: Intel Xeon E5504 2.00Ghz (4 cores)
- Memoria: 12GB
- Tarjeta de red: 2 x Intel Corporation 82574L Gigabit Ethernet
- Placa Base:

Como se puede ver estos equipos carecen de disco de almacenamiento por lo que se han empleado para como "músculo" del clúster para ejecutar todas las máquinas virtuales que se despliegan en este.

A parte de los equipos mencionados anteriormente también se dispone de un equipos con las siguientes características:

- CPU: Intel Xeon E5504 2.00Ghz (4 cores)
- Memoria: 12GB
- Tarjeta de red: 4 x Intel Corporation 82576 Gigabit Network Connection
- Controladora de discos: capaz de gestionar 16 discos SATA
- 10 discos sata de 1TB 7200 rpm

Esta última máquina al disponer de bastante almacenamiento y de 4 tarjetas de red se empleará como FrontEnd para OpenNebula y como servidor tftp y dhcp para el resto de máquinas del clúster.

2.2 Almacenamiento

Para el almacenamiento de datos, aprovechando las múltiples unidades de disco disponibles se planteó crear una estructura en raid con paridad para que las máquinas puedan acceder a toda la capacidad de almacenamiento y disponer de protección ante fallos.

Se plantean dos posibles soluciones, raid-5 y raid-6 con sus diferentes ventajas en inconvenientes:

RAID-5:

- Precio por GB de almacenamiento relativamente bajo
- Solo proporciona un punto de fallo entre todos los discos
- Capacidad de almacenamiento N-1

RAID-6:

- Coste del GB más alto que en RAID-5
- Proporciona 2 puntos de fallo antes del fallo total del almacenamiento
- Capacidad de almacenamiento N-2

Finalmente se ha optado por utilizar una infraestructura en raid-6 por las siguientes razones:

- La capacidad de disco final obtenida, 8TB, es suficiente para los requisitos de uso que se le quieren dar al clúster.
- Proporciona más tolerancia a fallos en uno de los componentes más propensos a fallar de una infraestructura junto con la placa base.
- La probabilidad de fallo de un disco SATA aumenta cuanto más grande es la capacidad del disco, tratando con discos de 1TB la probabilidad de fallo es lo suficientemente alta como para tener en cuenta un posible doble fallo en algún momento. En la figura2.1 se puede observar la evolución de la probabilidad de error en función de el aumento del tamaño de los discos durante los próximos años.

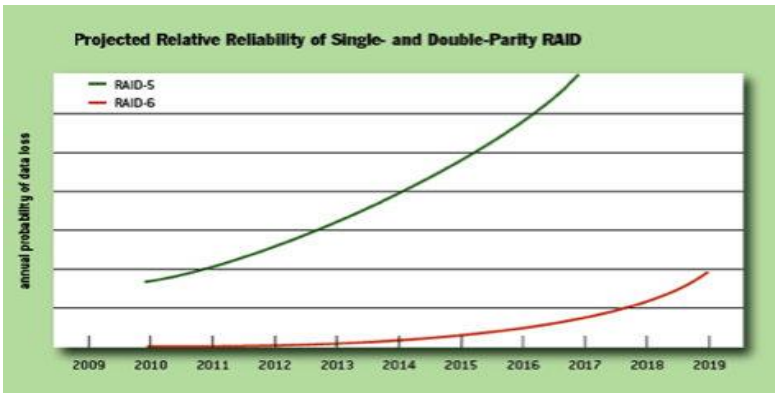


Fig 2.1 Evolución de la probabilidad de error en los próximos años [8]

2.3 Red

Toda la red del clúster la podemos dividir en dos redes distintas en según la función que desempeñan. Por un lado tenemos una red que se destina a proporcionar el arranque, el sistema operativo y los ficheros las máquinas hipervisor, mientras que la segunda es la encargada de proporcionar conectividad a internet, de manera que tenemos los dos servicios separados.
Además por temas de direccionamiento que se explicaran en un capítulo posterior se emplean dos VLAN en función del tipo de dirección (pública o privada) para gestionar el tráfico y su salida a internet.

Para conectar todas las máquinas hipervisor con el servidor y conectividad con el resto de la infraestructura de la empresa se emplea un switch de 24 puertos y capacidad para soportar GigabitEthernet.(SW1)

Este switch a su vez se conecta a al servidor DHCP “Poseidón” mediante otro switch que gestiona el tráfico en función de la VLAN a la que pertenezca. En la siguiente figura se puede ver el esquema de la red de forma simplificada.

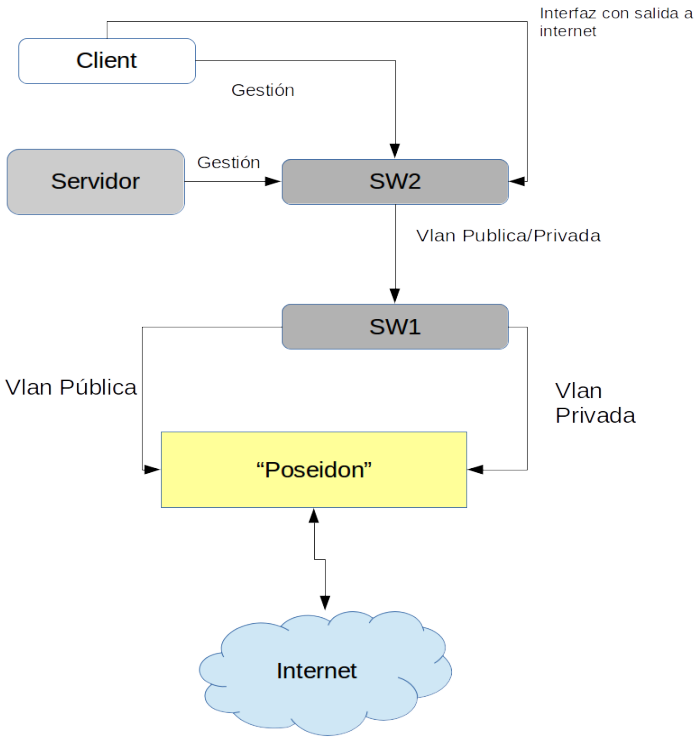


Fig 2.2 Esquema de red

CAPÍTULO 3 : PROCESO DE ARRANQUE

En este capítulo se explica todo el sistema de arranque del clúster. Este proceso es necesario para que todo el hardware del clúster pueda ejecutar el software base, kernel, sistema operativo y sistema de archivos sobre el que se desplegará la plataforma de explotación de recursos explicada en el apartado anterior. Para finalizar se explicarán los dos hipervisores utilizados durante el proyecto y los cambios a efectuar para integrarlos en todo el proceso.

3.1 Arranque

Durante el proceso de arranque de una maquina, esta pasa por un conjunto de fases sucesivas (ver fig 3.1) en las que cada una se encarga de preparar el entorno, y en las que además se va aumentando las funcionalidades y complejidad del conjunto.

Los datos y las funcionalidades más básicas están comprendidas dentro de los que se denomina el firmware del equipo (como por ejemplo las viejas BIOS o la más reciente UEFI). Este se encuentra dentro de la memoria de los chips de tipo persistente como la memoria ROM, EPROM o flash y se dedica a proporcionar funcionalidades y servicios a las capas superiores de software.

Una de mencionadas funcionalidades mentadas anteriormente es la "*first-stage boot loader*" que localiza, recupera, carga en memoria y transfiere el control al software de arranque. Un ejemplo de ello son soluciones como GRUB, LILO o PXELinux.0

Los dos primeros son ejemplos de "*first-stage boot loader*" ubicados directamente en el MBR (Master Boot Record) del dispositivo de arranque siendo este un HD, SSD, DVD etc. PXELinux.0 a diferencia del resto se encuentra en una maquina remota en la red local. Esta característica es la que nos permitirá arrancar el sistema operativo ya que es independiente a los dispositivos de almacenamiento de la maquina.

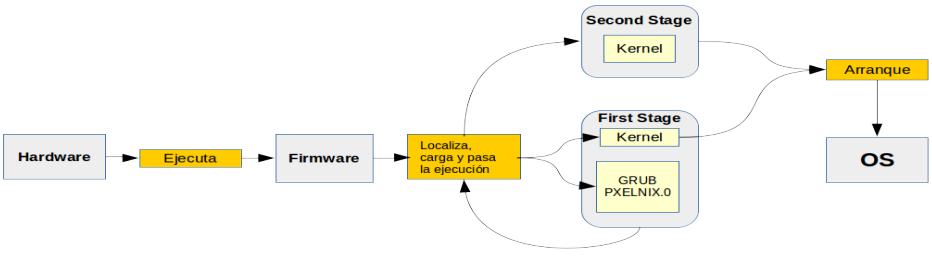


Fig 3.1 Esquema del arranque de un sistema operativo

3.1.1 DHCP(Dynamic Host Configuration Protocol)

DHCP es un protocolo de red para la distribución de configuraciones de red. Con DHCP daremos direccionamiento IP a las máquinas del clúster, asignaremos los hostnames y especificaremos el archivo pxelinux.0 para que los clientes soliciten el archivo por tftp al arrancar.

```
allow booting;
allow bootp;

subnet 192.168.2.0 netmask 255.255.255.0{
	range 192.168.2.2 192.168.2.10;
		option routers 192.168.2.1;
		filename "/pxelinux.0";

		host 1{
		hardware ethernet 00:30:48:b8:ac:98;
		option host-name "Sarah";
		fixed-address 192.168.2.3;
		}
		host 2{
		hardware ethernet 00:30:48:bc:7f:00;
		option host-name "Beth";
		fixed-address 192.168.2.4;
		}
		host 3{
		hardware ethernet 00:30:48:bc:7f:24;
		option host-name "Alison";
		fixed-address 192.168.2.5;
		}
```

Cuadro 3.1 Ejemplo configuración dhcp

En este archivo de configuración (fig 3.1) asignamos el rango de ips que asignará el servidor dhcp (del 2 al 10), el gateway por defecto (192.168.2.1) y el archivo de a transferir por tftp.
Para cada host configuramos tanto el hostname y la dirección del rango que le queremos asignar, para poder identificar a las máquinas utilizaremos su dirección MAC (hardware-address)

3.1.2 NFS (Network File System)

NFS es un protocolo a nivel de aplicación desarrollado por Sun MicroSystems, que permite utilizar un sistema de archivos distribuido por red. De esta manera los distintos sistema conectados pueden acceder a ficheros en un servidor remoto como si se tratase de archivos locales.

NFS está dividido en dos componentes, un servidor que es el que se encarga de brindar los archivos por red, y los clientes que simplemente acceden de manera remota a estos.

Todas los operaciones de NFS se realizan de manera síncrona. Por lo tanto, el servidor solo retorna una operación una vez se ha completado todo el trabajo. Por ejemplo en el caso de una operación de escritura el servidor primero escribirá todos los datos en el disco, actualizará la estructura de directorios si esta ha sufrido algún cambio y para finalizar devuelve la respuesta al cliente, de esta manera se garantiza la integridad de los datos.

El servidor NFS es necesario para poder proporcionar un sistema de archivos a las máquinas que arrancarán en el clúster las cuales actuarán como clientes NFS

3.1.3 TFTP (Trivial File Transfer Protocol)

TFTP es un protocolo para la transferencia sencilla de archivos similar a FTP. Se acostumbra a utilizar para enviar pequeños archivos entre maquina por red, como podría ser un kernel de Linux.
TFTP utiliza UDP (como protocolo de transporte) en el puerto 69
TFTP al ser un protocolo tan sencillo no soporta ni listado de directorios, ni autenticación ni cifrado.

En este proyecto se ha empleado TFTP para proporcionar por red el kernel, el initrd, y la imagen de xen a las máquinas que arrancan dentro del clúster. Necesario para que se realice de manera correcta todo el proceso de arranque.

3.1.4 PXE (Pre Execution Environment)

El protocolo PXE consiste en una combinación de los protocolos DHCP y TFTP con leves modificaciones en ambos. DHCP se utiliza para localizar el servidor DHCP mientras que con TFTP se descarga el programa inicial de bootstrap y archivos adicionales.

PXE se encarga de extender las funcionalidades del firmware para que de esta manera, una maquina pueda arrancar un sistema operativo a través de una red, y de forma independiente a sus dispositivos de almacenamiento.

El firmware del cliente trata de encontrar un servicio de redirección PXE por red, y de este modo encontrar los servidores PXE disponibles. Una vez analizada la respuesta, el firmware solicita el filepath de un NBP(network bootstrap program), y lo descarga en la memoria RAM de la máquina para posteriormente ejecutarlo.

Para iniciar una sesión de arranque con PXE el firmware envía un paquete del tipo DHCPDISCOVER con algunas extensiones específicas de PXE al puerto 67(el estándar de DHCP).

Cuando el servidor recibe un paquete DHCPDISCOVER este responde con un paquete DHCPOFFER (Extendido) que contiene:

1. la lista de direcciones IP de los servidores PXE
2. un menú en el que cada entrada corresponde a un servidor PXE
3. un prompt para que el usuario pueda ver el menú de arranque
4. y generalmente un tiempo de espera que en el caso de expirar lanzará la primera opción del menú de arranque.

Una vez se escoge una opción de arranque se envía una paquete del tipo DHCPREQUEST al servidor PXE, este paquete contiene el servidor de arranque PXE y la capa de arranque PXE, que será respondido por el servidor PXE con un paquete del tipo DHCPBACK extendido con la siguiente información:

1. el filepath completo para descargar el NBP vía TFTP
2. el tipo de servidor de arranque PXE y la capa de arranque PXE

3.1.5 SYSLinux

Syslinux es un conjunto de bootloaders y submódulos capaces de arrancar sistemas operativos Linux desde discos duros, CD o vía red. Soporta sistemas de archivos FAT, ext2, ext3, ext4, btrfs y NFS vía red empleando PXE

3.1.6 PXELinux.0

Pxelinux es un derivado de Syslinux para arrancar desde un servidor en red utilizando un ROM en red, PXE, como entorno de pre-ejecución.
Por lo tanto pxelinux.0 actúa como "*second-stage-bootloader*", permitiendo la ejecución del kernel de Linux y su correspondiente initrd.
PXE permite utilizar un archivo de configuración para parametrizar la ejecución del kernel así como configurar menús gráficos.

Cada cliente puede disponer de un fichero de configuración, el nombre del fichero vendrá determinado o por su dirección MAC o por su dirección IP convertida a hexadecimal. Si no se encuentra un fichero de comunicación asignado a esa máquina se procede a utilizar el fichero por defecto 'default'. Para este proyecto se ha decidido que todas las máquinas utilizarán el fichero default por defecto, pues todas van a realizar las mismas funciones.

```
Default Linux

LABEL Linux
```

```
Kernel mboot.c32
APPEND xen-amd64.gz --- vmlinuz-3.19.0-25-generic root=/dev/nfs nfsroot=
192.168.2.1:/nfsroot ip=dchp --- initrd.img-3.19.0-25-generic
```

Cuadro 3.2 Ejemplo archivo default

Una vez la maquina ha descargado el fichero de configuración este se interpreta.
La primera línea provoca que se cargue por defecto el LABEL Linux. La tercera línea provoca que se descargue por tftp el binario mboot.32.Este binario es un submódulo de syslinux y permite la ejecución múltiple y sincronizada de imágenes al arrancar el sistema. En nuestro caso de uso nos permitirá cargar el hipervisor XEN que depende de tanto del kernel como del initrd.
En la última línea se indica el orden en el que cargará las imágenes y los parámetros con lo que cargará, en este caso cargará primero el módulo de Xen, y posteriormente el kernel y el initrd.

En este caso solo está parametrizada la imagen del kernel de Linux, en el que se indica:

- root=/dev/nfs este parámetro es necesario para habilitar el pseudo-servicio NFS, para indicarle al kernel que utilice NFS en vez del disco para cargar el file system.
- nfsroot=<ip-servidor>:<directorio root> este parámetro indica al kernel de donde cargar el filesystem por NFS
- ip=dhcp indica al servidor que cargue el pseudo-servicio DHCP para obtener direccionamiento

Una vez se han cargado el kernel y el initrd en memoria. Pxelinux.0 transfiere el control de ejecución, la memoria donde se aloja el initrd y los parámetros de arranque al kernel de Linux.
Este vía NFS carga el sistema de archivos raíz, se descomprime a si mismo y extrae el initrd en el sistema de archivos raíz. Finalmente transfiere el control de ejecución al init.

En el siguiente esquema se puede observar paso a paso todo el proceso de arranque, desde que el cliente inicia el cliente DHCP para arrancar por red, hasta que ejecuta el initrd.

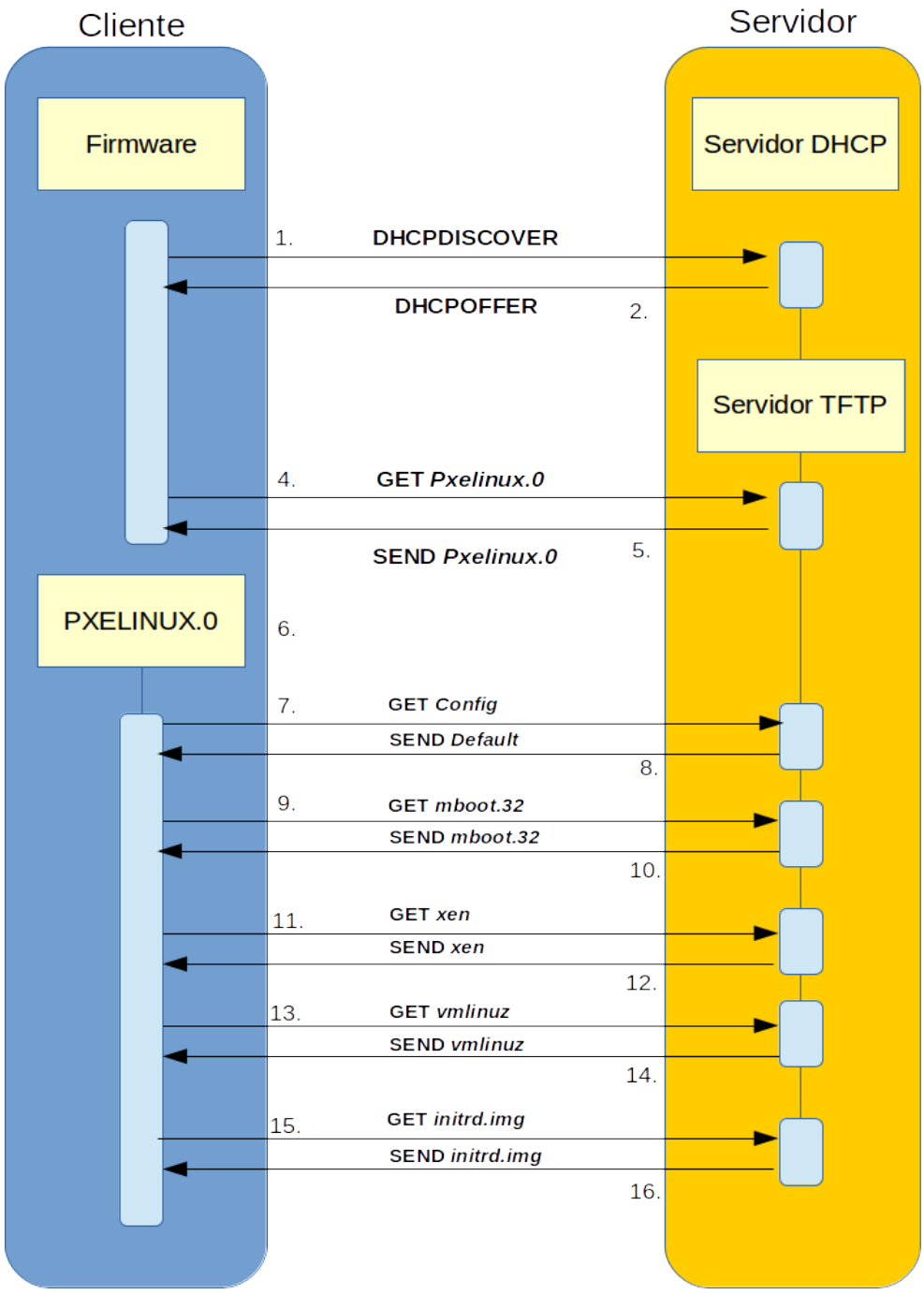


Fig 3.2 Esquema del proceso de arranque de una máquina del clúster

1. La maquina al arrancar por red busca un servidor DHCP disponible mediante un DHCPDISCOVER por broadcast.
2. El servidor DHCP le responde con un DHCPOFFER
3. La maquina se configura la interfaz de red en función de la información ofrecida por el servidor dhcp.

4. La maquina pide al servidor tftp el binario pxelinux.0 tal y como viene establecido en la configuración obtenida por dhcp
5. El servidor TFTP responde a la maquina enviando el binario
6. La maquina ejecuta pxelinux.0 y se inicia el "*second stage bootloader*".
7. La maquina solicita al servidor tftp su archivo de configuración.
8. Al no tener ningún archivo de configuración específico para esa IP o dirección MAC el servidor tftp envía el archivo por defecto *default.*
9. Pxelinux.0 solicita el archivo mboot.32 tal y como viene establecido en el archivo default
10. El servidor envía el archivo mboot.32 al cliente
11. La maquina solicita el archivo de xen
12. Envío del archivo xen
13. La maquina solicita la imagen del kernel de Linux *vmlinuz*
14. Envío del kernel
15. La maquina solicita el initrd.img
16. Envío del initrd.img

3.2 XEN Hipervisor

Xen es un hipervisor usando un diseño en microkernel que permite ejecutar múltiples instancias de un mismo sistema operativo o de diferentes sistemas operativos en una misma máquina utilizando su hardware de manera concurrente.

- Ocupa poco espacio(alrededor de 1MB de tamaño). Debido a que utiliza un diseño de microkernel, con una pequeña huella de memoria y una interfaz limitada para el huésped, es más robusto y seguro que otros hipervisores.
- La mayoría de las instalaciones funcionan con Linux como stack principal, conocido como dom0, pero también puede funcionar en BSD o OpenSolaris
- *"Driver Isolation"*: permite que el driver de la maquina huésped funcione en el sistema de una máquina virtual.
- Permite paravirtualización: El hipervisor permite ejecutarse en máquinas que no soportan las extensiones de virtualización.

3.2.1 Arquitectura de Xen

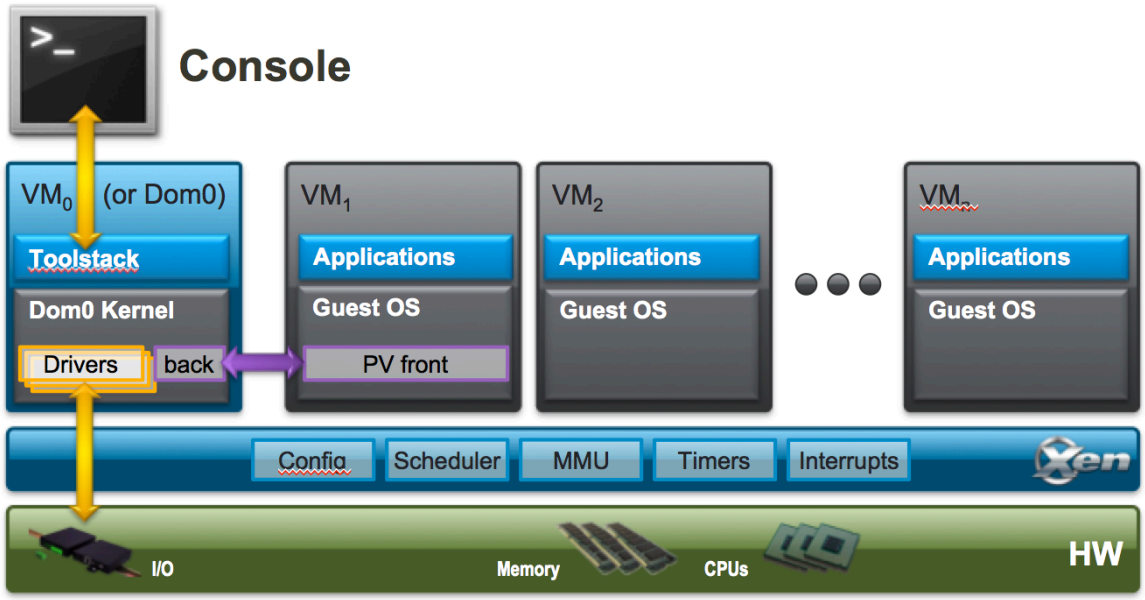


Fig 3.3 Esquema de la arquitectura de Xen [4]

Como se puede observar en la figura 3.3 Xen esta formado por diversos componentes entre los que podemos destacar:

VM (Virtual Machines) son entornos virtualizados ejecutando cada uno su sistema operativo y sus aplicaciones. El hipervisor puede funcionar en dos modos, paravirtualización (PV) y Hardware-assisted o Full-Virtualized (HVM). Ambos modos se pueden ejecutar en el mismo hipervisor. También es posible ejecutar máquinas PV dentro de máquinas HVM. Las máquinas virtuales no tienen acceso directo al hardware.

El **dominio de control** (Domain-0) es una maquina virtual con privilegios especiales, tiene acceso directo al hardware y se encarga de todas las operaciones I/O del sistema e interactúa con el resto de máquinas virtuales. Además proporciona una interfaz de control (toolstack) para gestionar el sistema. Xen no se puede utilizar sin esta máquina virtual que además es la primera en ejecutarse.

La **toolstack** permite gestionar la creación, destrucción o configuración de las maquinas virtuales. La toolstack expone una pila de control accesible de desde consola de comandos, interfaz gráfica o desde sistemas de computación en la nube como OpenNebula.

Dom-0 requiere un kernel habilitado para ejecutar Xen. La mayoría de distribuciones Linux basadas en las últimas versiones del kernel de Linux son capaces de ejecutar Xen ya que contienen los paquetes necesarios para su ejecución y el uso de la toolstack. Además para poder ejecutar máquinas en

modo PV el kernel también tiene que soportar paravirtualización. Solo las versiones viejas del kernel de Linux no soportan esta funcionalidad.

3.2.2 Creación de un nuevo initrd (XEN)

Para que las máquinas del clúster puedan ejecutar Xen sin ningún tipo de problema tendremos que cargar una serie de módulos del kernel cuando este inicialice el sistema. Para ello utilizaremos el initrd, ya que este archivo se carga en memoria durante el arranque del sistema y es de donde el kernel de Linux copia a un sistema archivos temporal para luego ejecutar el proceso init resultante.

Para poder cargar los módulos tendremos que crear un nuevo initramfs diferente al que viene por defecto en el sistema, que se enviará vía tftp en el arranque junto a la imagen de xen y el kernel de Linux, con los siguiente módulos.

- Xenfs: permite utilizar una versión modificada de NFS para que Xen pueda compartir archivos con una maquina xen.
- Xen-evtchn: carga un driver que permite un proceso de usuario para activar canales de eventos y para recibir notificaciones de los eventos que ocurren en estos
- Xen-netback: Xen Network Backend Driver, driver que permite a Xen comunicar la maquina anfitrión con la maquina virtualizada
- Xen-blkback: driver que proporciona una interfaz unificada capaz de ser usada por cualquier sistema operativo
- Xen-blktap: permite a los dispositivos virtuales presentes en las máquinas virtuales ejecutarse en espacio de usuario y ser respaldados por las particiones sin procesar, archivos y red
- Bridge: permite que las máquinas virtuales compartan la conexión de red con la maquina anfitrión

3.3 KVM (Kernel-Based Virtualization Machine)

KVM es una plataforma de virtualización creada para Linux sobre arquitectura x86 que soporte las extensiones de virtualización Intel VT o AMD-V. Consiste de el modulo del kernel de Linux kvm.ko, que es el que ofrece la infraestructura de virtualización. Y los módulos para procesadores de fabricantes específicos, kvm-intel.ko o kvm-amd.ko

KVM no realiza ningún tipo de emulación (para ello utiliza Qemu) en lugar de ello expone una interfaz que permite:

- Configurar el espacio de direcciones de una maquina virtual así como y una imagen de firmware(generalmente una BIOS) que la maquina virtual utiliza para arrancar su propio sistema operativo.
- Comunicarse con los canales I/O de las VM

3.3.1 Arquitectura de KVM

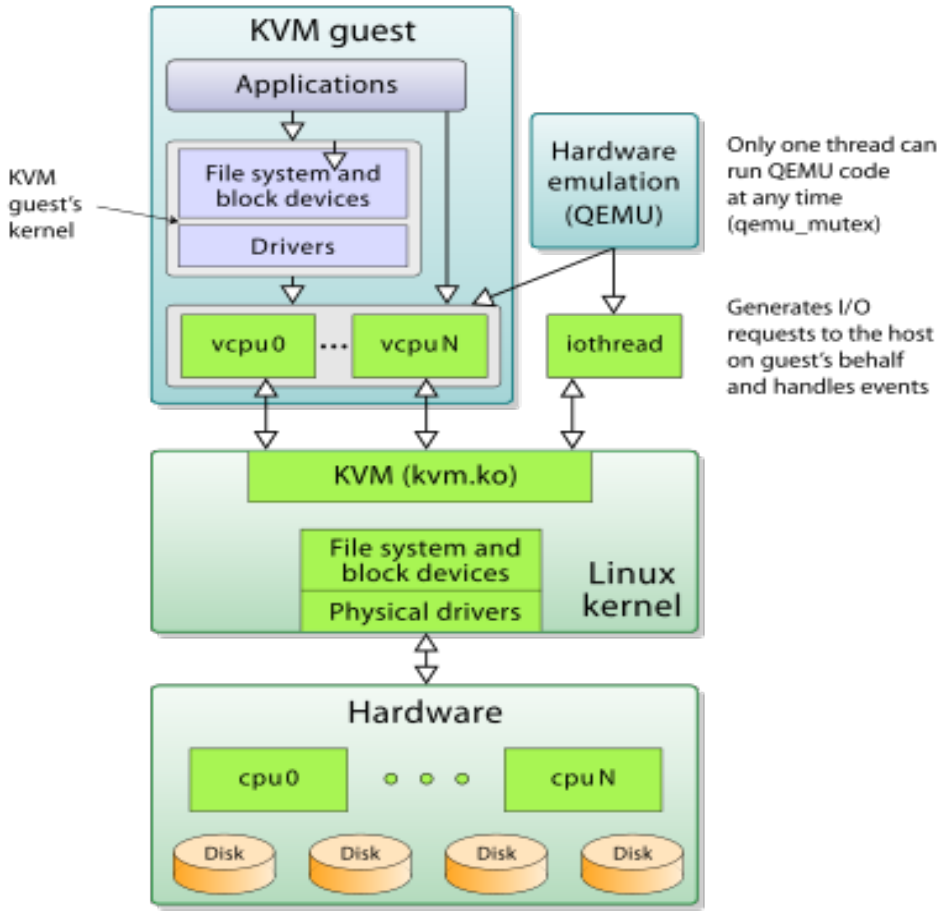


Fig 3.4 Esquema de la arquitectura de KVM [3]

KVM funciona como un módulo del kernel de Linux y se encarga de realizar principalmente 3 funciones:

- Comunicarse con el hardware físico de la máquina
- Ejecutar la maquina virtual
- Comunicarse con qemu

QEMU se encarga de emular el hardware que teóricamente tiene la maquina virtual, para realizar esto se comunica por un canal I/O directamente con el módulo de KVM (figura 3.4)

Por último la **maquina virtual**, se encarga de cargar los drivers del hardware emulado, y ejecuta su kernel, los procesos y aplicaciones valiéndose de este.

3.3.2 Initrd KVM

Al igual que con Xen, para que las máquinas del clúster puedan ejecutar KVM necesitaremos que el kernel genere un init que cargue los módulos necesarios para el uso de KVM:
En el archivo de configuración de módulos de initramfs-tools, que es una herramienta para generar este tipo de binarios, añadiremos los siguiente módulos:

- Kvm.ko: modulo principal de kvm
- Kvm-intel.ko: módulo para habilitar la virtualización nativa sobre procesadores Intel
- Kvm-amd.ko : módulo para habilitar la virtualización nativa sobre procesadores AMD

CAPÍTULO 4. PLATAFORMA DE GESTIÓN

En este capítulo se detalla cómo funciona OpenNebula como orquestador y plataforma de gestión, se explicará que ofrece, cuál es su arquitectura, que elementos la componen y cómo gestiona los recursos de un clúster

4.1 OpenNebula

OpenNebula es una plataforma de computación para centros de datos distribuidos y heterogéneos. OpenNebula ofrece una infraestructura para poder desplegar implementaciones de IaaS (Infrastructure as a Service).

OpenNebula se encarga de gestionar la red, almacenamiento, los servicios de virtualización, los recursos de hardware y la seguridad.

4.1.1 Que Ofrece OpenNebula

- Usuarios y Grupos: OpenNebula permite la gestión de usuarios en grupos para poder asignar cuotas de uso en los recursos ya sea en computación, red o almacenamiento.
- OpenNebula es compatible con varios hipervisores (entre ellos KVM y XEN), permite controlar todo el ciclo de vida de las maquinas virtuales así como, mantener varios hipervisores en la misma infraestructura.
- OpenNebula proporciona una interfaz de gestión centralizada para los host físicos (máquinas físicas que serán anfitrionas de las máquinas virtuales)
- OpenNebula monitorea de manera constante los parámetros clave para saber el estado y el rendimiento de las máquinas, de manera que se puede controlar y planificar los recursos de las máquinas mediante políticas.
- Un sistema de contabilidad configurable para visualizar y comunicar los datos de uso de recursos, para permitir su integración con las plataformas de facturación y cargos, o para garantizar la participación equitativa de los recursos entre los usuarios
- Un sistema para integrar de manera fácil las máquinas al centro de datos y para permitir el aislamiento de las máquinas virtuales del resto.
- Soporte para diferentes subsistemas de almacenamiento.
- Mecanismos de autenticación y autorización que permite que diversos mecanismos sean empleados para identificar a los usuarios, así como mecanismos de control de acceso.
- Soporte para infraestructuras de alta disponibilidad.

- Clusters: Un clúster es un conjunto de máquinas, recursos de almacenamiento y redes virtuales. Empleados para balanceo de carga, alta disponibilidad y alto rendimiento.
- Virtualización de Centros de datos.

4.2 Arquitectura de OpenNebula

OpenNebula asume que la infraestructura física sigue la arquitectura clásica de clúster (figura 4.1), con un frontend una serie de hosts y una red física que conecta todos los componentes.

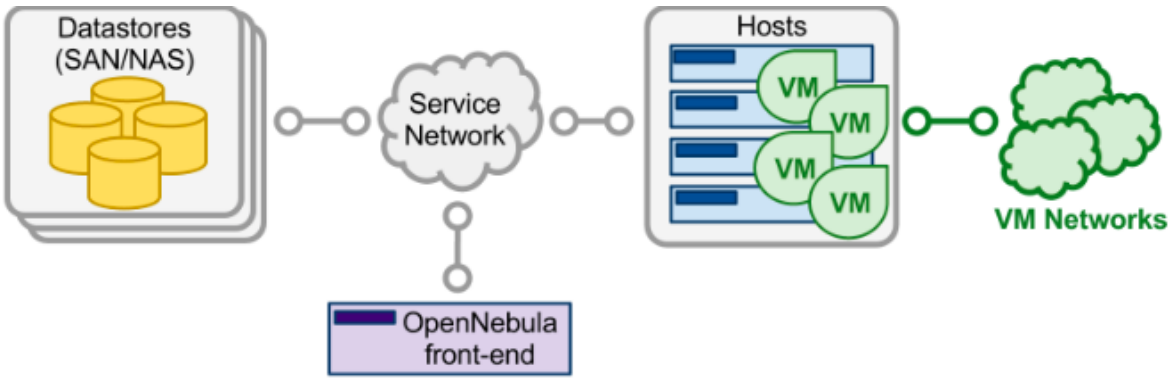


Fig 4.1 Esquema arquitectura OpenNebula [9]

Los componentes básicos de un sistema OpenNebula son:

- **Frontend**: es la maquina principal, ella se encarga de ejecutar todos los servicios de OpenNebula, la gestión de recursos, monitorear los hosts, la seguridad, el control de usuarios etc
- **Hosts**: son máquinas que ejecutan un hipervisor, se encargan de ejecutar el conjunto de máquinas virtuales que funcionan en toda la infraestructura.
- **Datastores**: se encargan de almacenar las imágenes base de las máquinas virtuales, en nuestro caso el datastore y el frontend son la misma máquina por lo que además contiene todos los datos de los hosts.
- Una **red** física operable que permite la comunicación de todos los dispositivos, indispensable para el funcionamiento.

4.2.1 Frontend

La maquina con la instalación del servicio principal de OpenNebula es llamada FrontEnd. Esta maquina necesita conectividad con todos los host de la plataforma y acceso a los datastores, en nuestro caso el frontend se encuentra

en la misma maquina que ofrece el almacenamiento por lo que en este aspecto no existe complejidad alguna.

El servicio de OpenNebula incluye:
- El daemon de gestión (oned) y el scheduler (mm_sched)
- El servidor web (opennebula-sunstone) que ofrece una interfaz web para la gestión de toda la plataforma.

4.2.2 Almacenamiento

OpenNebula utiliza los denominados Datastores para gestionar los discos virtuales de las diferentes VM. Típicamente los datastores se encuentran respaldados en un SAN/NAS. Generalmente todo datastore tiene que ser accesible por el frontend vía SAN o NAS (figura 4.2) o como ocurre en nuestro caso accede mediante almacenamiento conectado directamente.

Cuando una maquina virtual es desplegada en un host de la plataforma, la imagen es transferida de el datastore al host. Que dependiendo de la tecnología de almacenamiento se hará mediante una transferencia real, un enlace simbólico o creando un volumen virtual.

OpenNebula distingue 3 clases de datastores:

- **System Datastore**: este datastore se encarga de almacenar las imágenes de disco de las diferentes máquinas virtuales que se están ejecutando en la plataforma.
- **Image Datastore**: actúa como repositorio de las imágenes con las que se crearan las máquinas virtuales. Estas imágenes son movidas o clonadas en/desde el System datastore en función de si la VM es desplegada, apagada, o cuando se realizan snapshots de la imagen.
- **File Datastore**: en esta unidad de almacenamiento los ficheros planos y todo lo que no sea una imagen de disco como podrían ser kernels, ficheros de contexto.

A parte de estos 3 tipos de datastore, el Image Datastore puede funcionar en diferentes modos según la tecnología de almacenamiento que empleada:

- Sistema de ficheros, para guardar las imágenes en forma de fichero. Estos ficheros se almacenara en un directorio montado de el SAN/NAS
- VMFS un sistema de archivos especializado para ser utilizado con el hypervisor de VMWare
- LVMs, el driver de OpenNebula permite utilizar volúmenes lógicos en lugar de sistemas de ficheros tradicionales para almacenar las imágenes virtuales.
- Ceph, para almacenar imágenes de disco utilizando dispositivos de bloque Ceph

Por defecto OpenNebula trabaja en modo sistema de ficheros y es como se ha planteado que va a funcionar la plataforma desplegada en este proyecto.

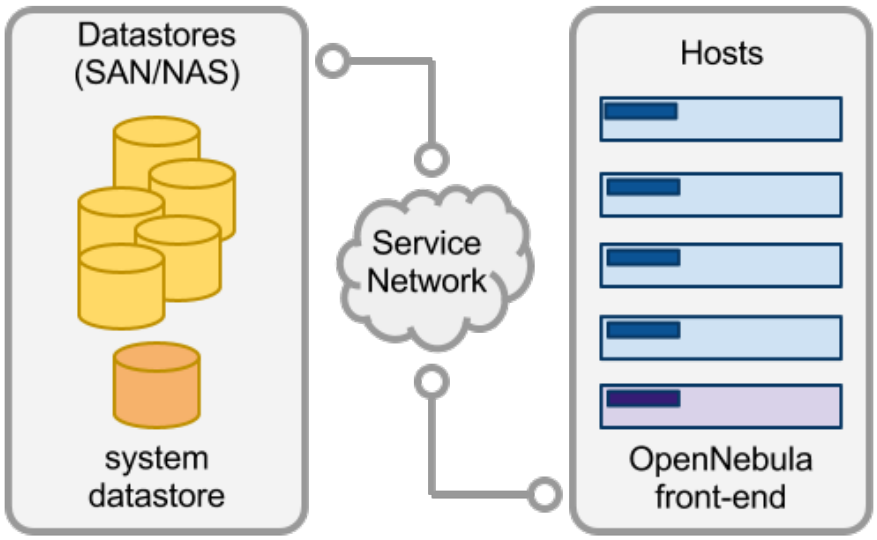


Fig 4.2 Comunicación datastore con el clúster [9]

4.2.3 Red

OpenNebula proporciona un sistema de red adaptable y personalizable para facilitar la integración de las máquinas virtuales con los requisitos de red del centro de datos, para ello se necesitan mínimo dos redes:

- La red de servicio: esta red es la que utiliza el frontend de OpenNebula para comunicarse con los nodos y acceder a ellos de manera que pueda gestionarlos, hacer tareas de monitorización y transferencia de datos e imágenes. Se recomienda que esta red esté separada del resto para evitar congestión y fallos.
- Una red para poder ofrecer conectividad a cada una de las máquinas virtuales alojadas en los hosts. Para ello es recomendable que esta red se localice en una red física diferente a la red de servicio.

Para el correcto funcionamiento es necesario asignar un driver de red a cada host en el momento de crearlos:

- Dummy: El driver por defecto, no realiza ninguna operación de red más allá del envío y recepción de paquetes.
- Fw: Este driver emplea diferentes reglas de firewall definidas por el administrador
- 802.1Q: restringe el acceso aplicando el uso de VLANs, para que este driver se pueda aplicar el hardware de red (switches) deben soportar este estándar.
- Ebtables: restringe el acceso de red aplicando reglas ebtables.
- Ovswitch: driver de red para emplear acceso mediantes Open vSwitch Virtual Switch
- VMWare: utiliza el driver de VMWare.

4.3 Recursos Virtuales

4.3.1 Imágenes

Se consideran imágenes dentro del sistema de almacenamiento de opennebula, tanto imágenes de sistema operativo, datos, o ficheros. Estos pueden ser utilizados por diferentes máquinas virtuales de forma simultánea.

Tipos de imagen:

- OS: esta imagen contiene un sistema operativo en funcionamiento. Cada template utilizado para crear una maquina virtual tiene que definir uno de sus discos apuntando a esta imagen.
- CDROM: esta imagen contiene datos solo de lectura. Solo se puede utilizar una por template, no son clonadas al utilizar almacenamiento compartido. Son útiles para generar imágenes de tipo OS con la ayuda de otra imagen de tipo datablock.
- DATABLOCK: imagen destinada al almacenamiento de datos. Estas imágenes pueden ser accesibles y modificadas por múltiples VM de manera simultánea. Este tipo de imágenes pueden ser creadas con datos previos o vacías de datos, como si de un disco se tratara.
- KERNEL: un archivo plano que será utilizado como kernel para las máquinas virtuales que lo tengan configurado en su template. Hay que destacar que a diferencia de los 3 tipos de imagen anterior que son almacenados en el image storage (OS, CDROM) y datablock (system storage), las imágenes de tipo kernel son alojadas en el File Datastore.
- RAMDISK: archivo plano utilizado como ramdisk para la maquina virtual que así lo solicite. Igual que las imágenes de tipo kernel estas también son alojadas en el File Datastore.
- CONTEXT: archivo plano utilizado como archivo de contexto para la maquina virtual que así lo solicite. También son alojadas en el File Datastore.

Ciclo de vida de una imagen:

1 - Lock: la imagen está siendo creada o copiada en el datastore
2 - Una vez se ha copiado por creado/copiado la imagen esta pasa al estado ready que indica que está disponible para ser utilizada
3- Si un usuario deshabilita una imagen esta pasaría a estado disable, en el proceso inverso la imagen volvería al estado ready.
4- Si la imagen está siendo eliminada del datastore pasaría al estado delete durante el proceso.
5- En caso de cualquier tipo de fallo, ya sea en a la hora de habilitar/deshabilitar o borrar la imagen entraría en estado de error.

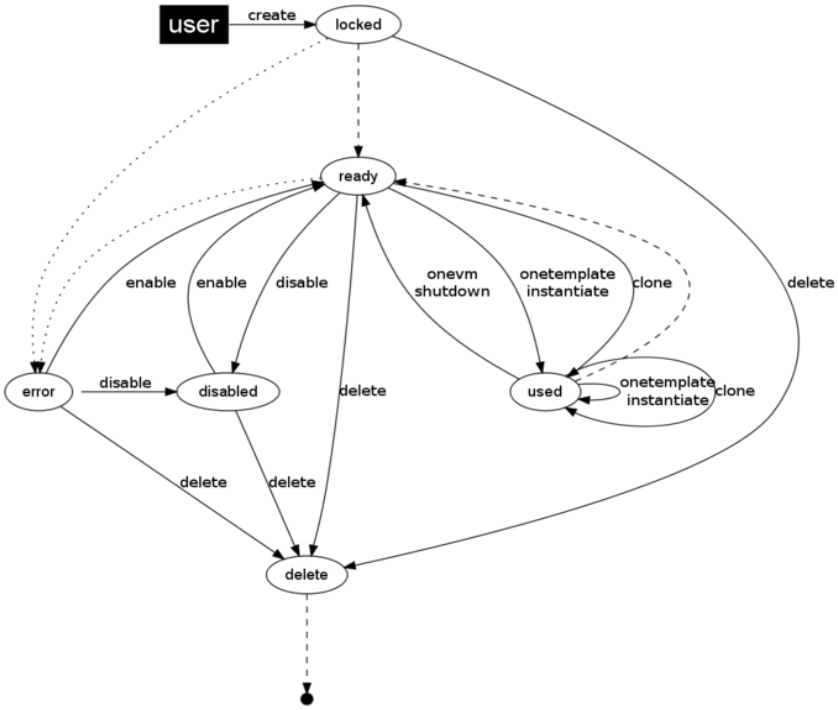


Fig 4.3 Ciclo de vida de una imagen [7]

Templates

Los templates son archivos parametrizados donde se define una plantilla a partir de la cual se instancian máquinas virtuales.
Si entendemos una maquina virtual como:

- Una capacidad determinada a nivel de memoria y CPU
- Un conjunto de imágenes de disco
- Un conjunto de interfaces de red conectadas a una o más redes virtuales

Para poder desplegar una maquina virtual que sea operativa el template tiene que satisfacer como mínimo los puntos mencionados anteriormente.

Definición de una template:

- Capacidad y nombre:
  - NAME: nombre para la maquina virtual
  - MEMORY: Cantidad de ram requerida por la VM (en megabytes)
  - CPU: Número de cpus que utilizará la VM (puede ser menos de 1 ej:0.5)
  - VCPU: Número de cpus virtuales que tendrá la maquina virtual.

- Disco:
  - IMAGE_ID: id de la imagen de disco
  - TYPE:
  - SIZE: tamaño de disco en MB
  - FORMAT: formato del sistema de archivos

- Red
  - NETWORK_ID: id de la red virtual
  - NETWORK_NAME: nombre de red virtual
  - NETWORK_UID: uid de la red virtual

```
NAME   = test-vm
MEMORY = 128
CPU    = 1

DISK = [ IMAGE  = "Debian" ]
DISK = [ TYPE     = swap, SIZE     = 1024 ]

NIC = [ NETWORK = "Public", NETWORK_UNAME="oneadmin" ]
```

Cuadro 4.1 Ejemplo template básico

Una vez tenemos creada la template podemos instanciar una maquina virtual mediante el uso de la template.

4.3.2 Virtual Machines

Estados de una maquina virtual desde que se crea hasta que se está ejecutando:

- **Pending**: es el estado inicial de una maquina virtual, por defecto inicializan en este estado mientras espera a que el FrontEnd OpenNebula le asigne una maquina y los recursos donde desplegarse
- **Prolog**: El sistema envía tanto las imagen y los archivos necesarios a la maquina que va a ejecutar la VM.
- **Boot**: OpenNebula espera mientras el hipervisor ejecuta la maquina virtual.
- **Running**: la maquina virtual se está ejecutando en el host

A parte de estos estados, el estado de la maquina virtual puede cambiar a otros en momentos puntuales:

- Migrate: El sistema está migrando la VM de un host a otro
- Hotplug: un disco o una NIC se ha agregado/quitado
- Save:Estado previo antes de una migración, parada o suspensión donde se guardan los archivos
- Snapshot: Se está ejecutando una snapshot de la VM
- Stopped: La VM esta parada

- Suspended: La VM está en estado de suspensión
- Undeployed: La maquina virtual esta apagada, además durante este estado los discos de la maquina virtual son transferidos al system datastore.
- Failed: La maquina virtual ha fallado en su despliegue, inoperativa.

4.3.3 Virtual Network

Para OpenNebula una Virtual Network es un elemento compuesto por tres partes diferenciadas:

- Una **estructura de red** que soportará la red virtual. Este elemento por lo general tendrá definidos los siguientes parámetros
  - BRIDGE: El bridge es una estructura lógica donde se vinculan la interfaz de red virtual de la Vm con la interfaz de red del host.
  - VLAN: si se activa la Virtual Network soportará tecnología VLAN para el aislamiento de redes locales.
  - VLAN_ID: identificador de VLAN

- Espacio de **direcciones lógicas** disponibles: Aquí se define el rango de IPs de las que dispondrá la red virtual. Este conjunto de direcciones puede ser o bien direcciones contiguas, o bien se pueden definir diferentes rangos de IPs (AR) para soportar direcciones no continuas.

- Atributos de **contexto**: En estos atributos se puede definir información de la red como podría ser:
  - la mascara de red
  - la puerta de enlace
  - los servidores DNS
  - la dirección IP de la red

```
NAME        = "Private Network"
BRIDGE = "bond-br0"
NETWORK_ADDRESS = "10.0.0.0"
NETWORK_MASK    = "255.255.255.0"
DNS             = "10.0.0.1"
GATEWAY         = "10.0.0.1"
AR=[
   TYPE = "IP4",
   IP   = "10.0.0.10",
   SIZE = "100",
]
AR=[
   TYPE = "IP4",
   IP   = "10.0.0.200",
   SIZE = "10",
```

Cuadro 4.2 Ejemplo de configuración de una red virtual

4.3.4 Esquema de red

La infraestructura de red empleada en el clúster se ha montado con el propósito de poder desplegar máquinas virtuales en dos virtual networks distintas:

- VN Privada: esta emplea direccionamiento privado mediante el uso del servidor DHCP “Poseidón” mencionado en capítulos anteriores, estas VM tienen que ser capaces de tener salida a internet, pero ser sólo accesibles desde la red empresarial
- VN Pública: en esta red se utilizan máquinas virtuales con dirección pública y por lo tanto tienen que ser accesibles desde cualquier terminal con acceso a internet.

Para poder diferenciar el tráfico de las dos redes virtuales se emplean dos vlans, la vlan 103 para todo el tráfico de la VN Privada, y la vlan 113 para el tráfico de la VN pública. Para ello se ha configurado cada virtual network de modo que su tráfico salga etiquetado con su respectiva vlan.

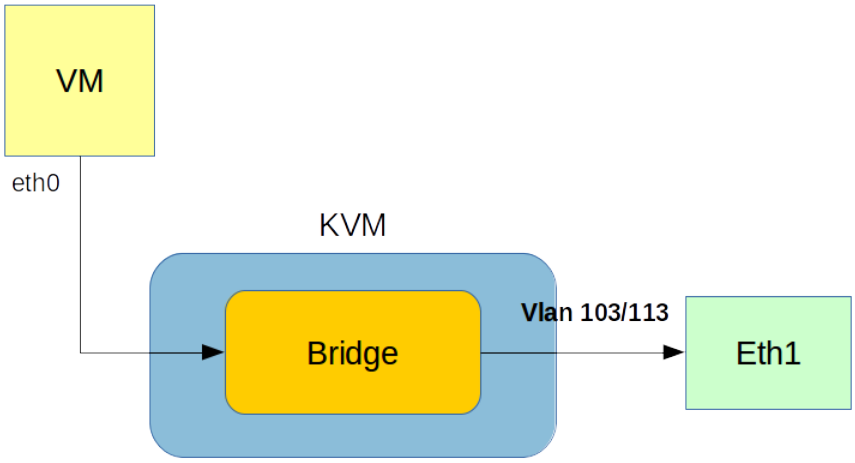


Fig 4.4 Esquema de comunicación entre una maquina virtual y una red mediante un bridge de red

El funcionamiento es sencillo, OpenNebula ejecuta una maquina virtual y configura su red en función de la virtual network a la que se ha asociado. Esta máquina virtual comienza a generar tráfico que es enviado a través de su interfaz de red.

OpenNebula genera un bridge de manera automática en la máquina anfitrión (ver figura 4.4) al cual todas las máquina virtuales se conectan. Este bridge actúa como intermediario entre la interfaz de red de la máquina virtual y la interfaz de red de la máquina física en la que se encuentra la virtual.

A parte de eso la máquina anfitrión al cargar el driver de red 8021Q de OpenNebula es capaz de etiquetar el tráfico en función de la VN de la máquina virtual, por lo que ya pasa el tráfico a la interfaz real etiquetado, de manera que esta solo tiene que hacer un forward hasta el switch que se encargará de gestionar los paquetes.

4.4 Escogiendo Hipervisor: Xen vs KVM

Durante el transcurso de este proyecto se han probado dos hipervisores Xen y KVM. En un principio la idea era utilizar XEN como hipervisor, pues es el sistema de virtualización utilizado en la empresa y con el que más experiencia cuentan los empleados. Sin embargo durante el transcurso del mismo han ido surgiendo complicaciones con este hipervisor y la solución propuesta que fueron alargando el tiempo estimado de implementación y que desembocaron en cambiar el hipervisor por KVM:

- El proceso de arranque es más complejo: se encontraron muchos problemas para poder conseguir que el clúster iniciara las máquinas y que estas pudieran ejecutar xen de manera funcional. Fue necesario emplear un sistema de boot múltiple y secuenciado, al que fue necesario agregar diversos módulos de los que no se tenía conocimiento de su función.
- Xen no es capaz de ejecutar máquinas virtuales del estilo readonly como podrían ser las de tipo CDROM de OpenNebula, debido a un bug en la librería libvirt que emplea, por lo tanto resulta imposible crear imágenes de SO a medida a partir de las ISO de los SO que se deseen.

Este último problema fue el que decantó la balanza en favor de KVM, el proceso de arranque fue más sencillo de realizar y se entendía mejor, y se pudieron crear imágenes propias.

Sin embargo surgió un pequeño problema, también relacionado con la librería libvirt, que se producía cuando se intentaba hacer una migración en caliente de una maquina virtual de un host a otro.

Por defecto libvirt consulta el uuid del sistema para identificar las máquinas que tiene intervienen en la migración, generalmente este uuid se genera en el arranque del sistema operativo leyendo los códigos del hardware empleado. Al ser todas las máquinas iguales el uuid empleado era el mismo y por lo tanto libvirt entendía que se intentaba migrar la VM a el mismo host y por lo tanto cancelaba el proceso.

Afortunadamente libvirt dispone de un fichero de configuración en el cual se puede indicar que uuid se desea emplear, por lo tanto el problema se pudo solucionar creando un script que cambia este uuid y que se ejecuta nada más arrancar el sistema operativo.

CAPÍTULO 5. INSTALACIÓN

En este capítulo se muestra paso a paso como instalar toda la plataforma de manera que al finalizar tendremos un sistema capaz de desplegar toda la plataforma de virtualización y gestión del clúster.

5.1 Requisitos

- Un sistema operativo Linux con un servidor nfs y un servidor tftp
- Al menos un cliente booteable con PXE
- Suficiente espacio en disco para alojar el sistema de ficheros de las máquinas cliente
- Conectividad entre máquinas
- Un servidor DHCP que soporte clientes PXE
- Máquinas con capacidad de arranque por red

En este proyecto la instalación se ha realizado sobre la versión LTS 14.04 de ubuntu server y la siguiente explicación da por hecho que se utilizar dicha distribución, derivada, o similar.

5.2 Plataforma de arranque

El primer paso es realizar la instalación de los paquetes básicos que vamos a necesitar:

- Servidor dhcp: en nuestro caso el paquete isc-dhcp-server
- Servidor tftp: paquete tftpf-hpa
- Servidor nfs: nfs-kernel-server
- Herramienta para creación de ramfs: initramfs-tools
- Syslinux

5.2.1 Configuración del servidor dhcp

Como se ha explicado en capítulos anteriores es necesario un servidor dhcp para poder el archivo de boot pxelinux.0 a las máquinas y para la asignación de dirección ip y hostname. Para ello editaremos el fichero dhcpd.conf localizado en el directorio /etc/dhcp/

```
allow booting;
allow bootp;

subnet 192.168.2.0 netmask 255.255.255.0{

        range 192.168.2.3 192.168.2.10;
                option routers 192.168.2.1;
                filename "/pxelinux.0";

                host 1{
                hardware ethernet 00:30:48:b8:ac:98;
                option host-name "Sarah";
                fixed-address 192.168.2.3;
                }
                host 2{
                hardware ethernet 00:30:48:bc:7f:00;
                option host-name "Beth";
                fixed-address 192.168.2.4;
                }
                host 3{
                hardware ethernet 00:30:48:bc:7f:24;
                option host-name "Alison";
                fixed-address 192.168.2.5;
                }
                host 4{
                hardware ethernet 00:30:48:bb:97:1a;
                option host-name "Cosima";
                fixed-address 192.168.2.6;
                }
}
```

Cuadro 5.1 Archivo de configuración del servidor DHCP

Como se puede apreciar en la figura primero es necesario configurar el rango de direcciones que brindará el servidor dhcp a el resto de máquinas, en el caso de este clúster el servidor dhcp solo brindará el número de IPs necesarias para las máquinas desplegadas. Además por temas de conectividad entre las máquinas y el frontend de OpenNebula asignaremos a este último como la puerta de enlace por defecto.

El siguiente paso no es necesario pero si recomendable, asignaremos a cada maquina una IP y un hostname de manera fija, esto será de utilidad para facilitar la administración, como por ejemplo saber que maquina ha tenido un error desde el panel de opennebula, donde está desplegada determinada maquina virtual etc.

5.2.2 Generación de la imagen initrd

Como se mencionó en capítulos anteriores es necesaria la generación de un nuevo disco de ram por los diversos motivos:

- Permitir el arranque por red del sistema
- Permitir el montaje del sistema de archivos de la máquina mediante NFS

- La carga de diferentes módulos del kernel de Linux para el correcto funcionamiento tanto de los hipervisores como de OpenNebula.

Antes de generarlo es necesaria la edición de dos ficheros.

Initramfs.conf

El primer archivo a editar es el initramfs.conf localizado en el directorio */etc/initramfs-tools* donde nos aseguraremos que los siguiente flags estén activos de la siguiente manera.

```
BOOT=nfs
```

Cuadro 5.2 Flag "BOOT"

Con este cambio permitimos que el arranque se produzca via NFS y no desde un dispositivo de almacenamiento local como podría ser un disco duro o disco de estado solido.

```
MODULES=netboot
```

Cuadro 5.3 Flag "MODULES"

Este cambio es necesario para que se añadan los módulos básicos y los de red, y se esquive los block devices que representan a discos duros.

5.2.2.1 Modules

Localizado también en el directorio */etc/initramfs-tools* este archivo le indica a initramfs-tools que módulos debe agregar a parte de los base (necesarios para la ejecución de un sistema) y los de red.

```
kvm.ko
kvm-intel.ko
bridge
loop max_loop=255
8021q
```

Cuadro 5.4 Configuración archivo modules

Los módulos ***kvm.ko*** y ***kvm-intel.ko*** son necesarios para la carga de kvm como hipervisor y para añadir soporte de aceleración por hardware a su ejecución (esta característica tiene que ser soportada por el procesador de la máquina).

El módulo ***bridge*** es necesario para que el sistema soporte bridging en las diferentes interfaces de red, indispensable para que kvm arranque una máquina virtual y además necesario para poder ofrecer conectividad a las distintas máquinas virtuales, ya que estas salen a internet a través de la interfaz física de la máquina donde están siendo ejecutadas.

El módulo ***8021q*** es necesario para que la máquina soporte el estándar 802.1q que ofrece VLANs y el etiquetado de estas. En este caso es necesario pues las distintas máquinas virtuales son asignadas a una VLAN que tiene que ser etiquetadas para el correcto funcionamiento de la red, sin este módulo no podríamos ofrecer conectividad con direccionamiento privado y público con la configuración actual de red de la empresa

Una vez editados los ficheros se puede proceder a la generación del archivo con el siguiente comando.

```
Mkinitramfs initrd.img-`uname -r`
```

Cuadro 5.5 Comando para la generación de un archivo initrd

Una vez generado el archivo lo copiaremos dentro del directorio del servidor tftp */var/lib/tftpboot/* junto con el kernel, generalmente llamado vmlinuz seguido del número de la versión, localizado en el directorio */boot.*

5.2.3 Configuración servidor NFS

El siguiente paso es crear un directorio que contenga todos los archivos del sistema operativo que ejecutarán las máquinas del cluster. En este proyecto se ha proporcionado la distribución Ubuntu server en su versión 14.04 por ser una de las recomendadas por OpenNebula (junto a CentOS).

La imagen la podemos obtener por diferentes métodos como un archivo comprimido, pero en este se ha empleado la herramienta debootstrap que permite descargar toda la imagen de un sistema operativo dentro de un directorio de manera sencilla indicando distribución y arquitectura.

```
debootstrap --arch amd64 trusty /nfsroot
```

Cuadro 5.6 Comando de uso debootstrap

Una vez obtenida la imagen del sistema operativo que se quiere distribuir es necesario que el directorio sea exportado vía nfs, para ello editaremos el siguiente fichero.

5.2.3.1 /etc/exports

En el fichero exports se configura la lista de acceso de los diferentes sistemas de archivos que serán exportados por el servidor nfs a los clientes.

```
/nfsroot      192.168.2.0/24(rw,no_root_squash,no_acl,no_subtree_check)
```

Cuadro 5.7 Importación del directorio raíz del SO

- Lo primero de todo es especificar el directorio que queremos exportar
- Segundo indicar que máquina o máquinas pueden acceder a el. En el ejemplo de la figura permite el acceso a cualquier maquina de la red local 192.168.2.0, en caso de que sea indiferente la procedencia de la máquina podemos indicarlo con un *
- Rw: con este flag se permite la lectura y escritura en el sistema de archivos
- No_root_squash: permite que el sistema de archivos exportado se pueda montar como sistema de archivos raíz de la maquina cliente.
- No_acl: permite no mostrar la lista de acceso a las máquinas cliente
- No_subtree_check: esta opción no es necesaria, pero incrementa la velocidad con la que se exporta el sistema de archivos via NFS.

Una vez editado el fichero podemos exportar el directorio o bien reiniciando el servicio o con el comando *exportfs.*

5.2.4 Configuración del servidor TFTP

El primero paso a realizar es copiar los siguientes ficheros de la carpeta /usr/lib/syslinux a el directorio principal del servidor tftp localizado en /var/lib/tftpboot dentro de este se procederá a la creación del directorio pxelinux.cfg y dentro de este crear el archivo default.

```
DEFAULT linux
LABEL linux
KERNEL vmlinuz-4.2.0-27-generic
APPEND root=/dev/nfs initrd=initrd.img-4.2.0-27-generic nfsroot=192.168.2.1:/nfsroot ip=dhcp rw
```

Cuadro 5.8 Configuración por defecto del servidor TFTP

En la etiqueta KERNEL se indica el kernel que se desea enviar via PXE a los clientes, es imprescindible que esté en el directorio del servidor tftp.

En la etiqueta APPEND se añaden el resto de flags con la información necesaria:

- Root: localización del sistema root, como en este caso este se entrega mediante NFS es necesario indicarlo con */dev/nfs.*
- Initrd: aquí se indica el initrd a enviar junto con el kernel.
- Nfsroot: en caso de enviar el sistema de archivos via NFS se tiene que indicar la ip del servidor y el sistema exportado que se desea.
- Ip: cómo obtendrá dirección la máquina, en este caso se indicará via dhcp, pero se podría indicar de manera manual

Una vez generada la plantilla PXE para todas las máquinas es necesario modificar los permisos del directorio del servidor tftp (por defecto */var/lib/tftpboot*) para evitar obtener un error del tipo “File Not Found” o “Permission Denied”

5.2.5 Configuración FSTAB en los clientes.

Para que el sistema operativo del cliente monte los distintos sistemas de ficheros es necesario editar el archivo que configura la tabla de sistema de archivos */etc/fstab*

```
# /etc/fstab: static file system information.
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>

proc    /proc   proc    defaults        0       0
/dev/nfs        /       nfs     defaults        0       0
192.168.2.1:/var/lib/one        /var/lib/one    nfs     auto    0       0
```

Cuadro 5.9 Configuración archivo fstab

Se montará el sistema de archivos obtenido como root vía NFS en el punto de montaje / para que sea utilizado como sistema de archivos base
Es importante para más adelante montar via nfs el directorio /var/lib/one del servidor en el directorio /var/lib/one de los clientes, pues será necesario para el funcionamiento de OpenNebula.
Por último montaremos el sistema de archivos virtual /proc para acceder a información del hardware.

5.2.6 Configuración de red

Al arrancar via red el cliente ya ha realizado un DHCP discover y ha obtenido configuración antes de que se ejecute el sistema operativo. Por lo tanto es necesario que el sistema operativo no vuelva a intentar configurar otra vez dicha interfaz. Para ello se editará el archivo de configuración */etc/network/interfaces* indicando que la configuración de dicha interfaz se hará de forma manual.

```
iface eth0 inet manual

iface eth1 inet manual

auto xenbr0

iface xenbr0 inet dhcp
      bridge_ports p50p1
```

Cuadro 5.10 Configuración archivo /etc/network/interfaces

Además de eso se creará un **bridge** de red en la segunda interfaz de red del cliente (la que está conectada a la red con acceso a **internet**) de manera que obtenga direccionamiento mediante un servidor dhcp que ofrece ip y enrutamiento a internet. En este caso se **evita conflicto** con el anterior servidor DHCP pues este se ha configurado para que solo ofrezca dirección ip a determinadas direcciones **MAC** (las de las interfaces eth0).

NOTA: dependiendo del sistema operativo exportado vía NFS es posible que no esté instalado el paquete que ofrezca esta funcionalidad. En este caso se puede instalar desde el servidor haciendo chroot sobre el directorio (en este caso */nfsroot*) e instalar el paquete necesario, en Ubuntu server corresponde a *bridge utils*.

5.2.7 Modules de Linux

Para que el sistema cargue los módulos que hemos indicado en el initrd, es necesario que estos se encuentren en el sistema de archivos raíz. Como se ha empleado debootstrap como método para obtener el SO, este viene sin ningún modulo mas allá de los básicos, por ello es necesario pasarlos al nfsroot.

En este caso al tener tanto servidor como cliente la misma versión de sistema operativo este paso es tan fácil como copiar el directorio */lib/modules/"version del kernel"* al directorio con el mismo nombre dentro del nfsroot.

Una vez realizados estos pasos, el resultado es un conjunto de servidores que arrancan vía red, con un sistema de archivos cargado vía red, acceso a internet y un hipervisor que permite la creación de máquinas virtuales. Para gestionar este clúster se instalará OpenNebula como orquestador de manera que se pueda gestionar de manera centralizada.

5.3 Instalación y configuración de OpenNebula

Como se ha detallado en el capítulo anterior en OpenNebula se distinguen dos roles, por un lado está el FrontEnd que ejecuta todos los servicios de OpenNebula, y los nodos que son los encargados de ejecutar las máquinas virtuales. En este apartado se detalla la configuración del frontend de OpenNebula para el funcionamiento en el clúster montado en los anteriores.

El primer paso es añadir el repositorio de OpenNebula en nuestro sistema e instalar el paquete de *opennebula* y si se desea *opennebula-sunstone* para disponer de una interfaz web para realizar la gestión.

Una vez hecho se instalará el paquete *opennebula-node* en los clientes

5.3.1 Configurar NFS

Para que los nodos puedan acceder a las máquinas virtuales, los datastores, los ficheros y que el frontend pueda acceder a los nodos es necesario exportar vía nfs el directorio del servidor de OpenNebula */var/lib/one* al resto de nodos. Para ello se añadirá este directorio al archivo */etc/exports* mencionado con anterioridad.

```
/var/lib/one    192.168.2.0/24(rw,sync,no_subtree_check,no_root_squash)
```

Cuadro 5.11 Exportar directorio de OpenNebula en el archivo exports

5.3.2 Configurar las claves SSH

OpenNebula necesita autenticarse entre frontend y los nodos de manera que no necesite contraseña para acceder. Para ello se gestionará un par de claves RSA que se utilizaran para realizar autenticación vía SSH.

Antes de todo se tiene que aclarar que toda la gestión interna de OpenNebula, es decir todas las ejecuciones de los servicios y órdenes que se dan a los nodos son gestionadas por el usuario oneadmin, creado en el proceso de instalación, y cuyo directorio *home* es el propio directorio del servidor opennebula, es decir, */var/lib/one*.

Primero se copiaran las claves ssh localizadas en la home del usuario oneadmin y se copiaran dentro del directorio de claves autorizadas.

```
cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```

Cuadro 5.12 Incluir la clave rsa dentro de las claves autorizadas

Además se ha de crear el archivo de configuración ssh de este usuario para que no añada las los host al archivo known_hosts del sistema.

~/.ssh/config

```
Host *
StrictHostKeyChecking no
UserKnownHostsFile /dev/null
```

Cuadro 5.13 Archivo de configuración ssh para el usurario oneadmin

Por último se modificarán los permisos de archivo para que solo tenga acceso oneadmin, el usuario propietario.

5.3.3 Reconocimiento de los nodos

OpenNebula gestiona y distingue los nodos en función del nombre que se le asigne, para que los reconozca se puede hacer de diversas maneras, una podría ser mediante la configuración de un servidor DNS. Pero en este clúster, al estar todo en la misma red local, y ser una infraestructura que no está prevista de grandes cambios de máquinas ni de su disposición en la red, se ha optado por editar el archivo */etc/hosts* de manera que se le asignará un nombre a una determinada ip.
Para facilitar la administración los nombres corresponden a los hostnames asignados vía dhcp.

```
127.0.0.1       localhost
127.0.1.1       ProjectLEDA
192.168.2.3     Sarah
192.168.2.4     Beth
192.168.2.5     Alison
192.168.2.6     Cosima
192.168.2.7     Helena
192.168.2.8     Rachel
192.168.2.9     Krystal
192.168.2.10    Mika
```

Cuadro 5.14 Contenido archivo hosts

5.3.4 Añadir los nodos al FrontEnd

Antes de poder ejecutar una maquina virtual en el clúster, primero se tienen que registrar los nodos, como nodos en funcionamiento para OpenNebula.

onehost create “nombre del nodo” -i kvm -v kvm -n 8021Q

Cuadro 5.15 Comando para añadir un nodo a OpenNebula

Para ello se indica:

- El nombre del nodo
- El tipo de driver para el manager de información, necesario para la correcta monitorización, en este caso KVM
- El driver del hipervisor que ejecutará las máquinas virtuales en ese nodo.
- El driver de red, como se ha mencionado anteriormente las máquinas virtuales pertenecerán a una vlan u otra dependiendo del tipo de dirección asignada por lo tanto en este caso es necesario el driver 8021Q.

Una vez registrados todos los nodos, podemos listarlos y comprobar su estado.

```
oneadmin@ProjectLEDA:~$ onehost list
  ID NAME            CLUSTER   RVM      ALLOCATED_CPU      ALLOCATED_MEM STAT
   8 Sarah           -           0       0 / 400 (0%)     0K / 11.7G (0%) on
   9 Beth            -           0       0 / 400 (0%)     0K / 9.8G (0%) on
  10 Alison          -           0       0 / 400 (0%)     0K / 11.7G (0%) on
  11 Cosima          -           0       0 / 400 (0%)     0K / 11.7G (0%) on
  12 Helena          -           0       0 / 400 (0%)     0K / 11.7G (0%) on
  13 Rachel          -           0       0 / 400 (0%)     0K / 11.7G (0%) on
  14 Krystal         -           0       0 / 400 (0%)     0K / 11.7G (0%) on
  15 Mika            -           1     100 / 400 (25%) 1024M / 11.7G (8%) on
```

Fig 5.1 Lista de hosts gestionados por OpenNebula

Una vez se registran todos los nodos en OpenNebula, ya disponemos de un clúster completamente funcional con opennebula.

CAPITULO 6. PLANIFICACIÓN

Este capítulo mostrará las diferentes tareas que se han realizado durante el proyecto para conseguir los objetivos propuestos.

Estudio previo: en esta etapa se realizó un proceso de autoformación en las tecnologías a emplear durante el proyecto y se analizaron los objetivos a cumplir.

Diseño: En esta etapa se diseñó la solución a aplicar y se definieron los pasos a seguir.

Implementación: la fase mas larga en la que se implementó todo el proyecto tanto en entornos de pruebas como en el clúster de la empresa.

Testeo: pruebas realizadas al final de la implementación

Documentación: redacción de la memoria.

6.1 Tiempo de dedicación

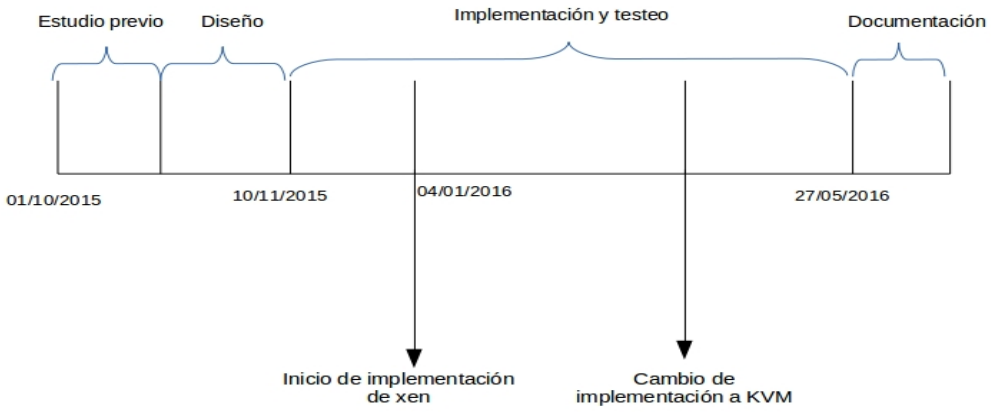


Fig 6.1 Planificación del proyecto

Como se puede observar en la figura anterior el tiempo de la implementación de la primera fase del proyecto es mucho mas grande en proporción al resto, esto se debe a varios factores. El desconocimiento de primeras de las tecnologías aplicadas provocaba que se tuvieran que realizar cambios contantemente en la forma de implementación, también como se explica en capítulos anteriores el uso de xen como hipervisor supuso un conjunto de problemas difíciles de resolver dada la poca documentación de estos problemas. Una vez se efectuó el cambio a KVM se comenzaron a obtener progresos mas rápidamente.

6.2 Tareas realizadas

En la siguiente tabla se puede observar el conjunto de tareas realizadas a lo largo del proyecto. Como se ha mencionado con anterioridad la implementación de Xen como hipervisor supuso un gran número de horas invertidas de ahí la duración tan larga del proyecto 740 h.

Tabla 1 Tareas realizadas

Tipo de tarea	Descripción	Tiempo
Estudio previo	Estudio de requisitos y búsqueda de información	40 h
Estudio previo	Pruebas de servidores DHCP y NFS	20 h
Estudio previo	Estudio de PXE	10 h
Diseño	Pruebas de uso de OpenNebula a nivel local	10 h
Diseño	Diseño de la infraestructura a montar	10 h
Diseño	Diseño del proceso de arranque	20 h
Implementacion	Desarrollo de fichero default pxe de una máquina básico	20 h
Implementación	Implementación del montaje del sistema de archivos via NFS	40 h
Implementación	Implementación de una máquina arrancando 3 módulos en multiboot	80 h
Implementacion	Implementación de una máquina arrancando Xen de manera funcional	240 h
Implementación	Instalación y configuración de OpenNebula	40 h
Testeo	Pruebas de funcionamiento (resultados deficientes)	30 h
Implementación	Cambios de configuración del cluster para utilizar KVM en lugar de Xen	30 h
Implementación	Cambios en la configuración de OpenNebula para cambiar de hipervisor	30 h
Implementación	Configuración vlans en la red	20 h
Testeo	Pruebas de funcionamiento	20 h
Documentación	Redacción de la memoria	80 h

CAPÍTULO 7. CONCLUSIONES

En este capítulo se intentará sacar unas conclusiones acerca del resultado del proyecto, en un primer lugar analizaremos que se ha realizado durante el transcurso del proyecto y si se han cumplido los requisitos expuestos en el capítulo 1. También se analizará el desarrollo a nivel personal haciendo énfasis en las aptitudes adquiridas. Seguido esto se finalizará mencionando posibles mejoras de proyecto en el futuro y una pequeña mención del impacto medioambiental del proyecto.

7.1 Conclusiones del proyecto

El propósito de este proyecto ha consistido en desplegar una plataforma de explotación para una empresa de manera que esta sea eficiente en cuanto al uso de recursos y permitiendo la gestión de manera centralizada. Ayudando así a resolver una problemática de trabajo de la empresa.

Para el desarrollo de este proyecto, ha sido necesario en primer lugar evaluar el modo de trabajo de la empresa y definir las necesidades de la empresa. Una vez definidas las necesidades de la empresa se pensó una solución que satisficiera todas las necesidades con el hardware disponible. Una pensada la solución se estudió qué tecnologías podrían utilizarse para su implementación. Este análisis se encuentra disponible en el capítulo 1.

Una vez pensada diseñada la solución y escogida la tecnología empleada se estudió el uso que se le daría al hardware disponible y a preparar la infraestructura física es decir, preparar el sistema de almacenamiento y la red. Capítulo 2

Una vez preparado el hardware se separó el desarrollo en dos partes, primero se implementa un sistema en el cual todas las máquinas arrancarían obtendrían su configuración para acabar ofreciendo un sistema de virtualización completo. Este proceso dadas la característica especial del hardware (almacenamiento local no disponible) es algo complejo y está explicado en el capítulo 3.

Después de tener un conjunto de máquinas arrancadas y funcionales, en el capítulo 4, se pasa a detallar el funcionamiento de la plataforma superior que gestionará el uso de este clúster y detalles de implementación específicos de este proyecto.

Para finalizar en el último capítulo es explica cómo implementar las dos partes explicadas y detalladas en los capítulos anteriores no tanto desde un punto de vista de arquitectura y funcionamientos sino técnico.

En cuanto a los objetivos asimilados, se puede decir que todos han sido cumplidos de manera satisfactoria.

Por una lado tenemos los **requisitos de gestión** de recursos tanto físicos como virtuales, podemos afirmas que ha sido **conseguido** pues OpenNebula permite gestionar de manera fácil la creación, migración y configuración de red de las maquinas virtuales. Como la capacidad de controlar tanto la creación de instancias y migraciones de virtuales en función de la carga de trabajo de las diferentes máquinas.

Los requisitos a nivel de gestión de **perfiles** han sido cubiertos pues opennebula permite crear y gestionar perfiles de usuario asignando permisos y cuotas de uso de los recursos.

OpenNebula proporciona una interfaz web llamada sunstone que ofrece información de clúster de manera visual, así como facilita la realización de tareas a los administradores. Además permite de un simple vistazo observar el estado de las máquinas los servidores, la carga de trabajo ya que plasma toda la información que opennebula monitoriza. Por lo tanto los requisitos de **monitorización** e interfaz **web** han sido asimilados de manera **satisfactoria**.

7.2 Conclusiones personales

Gracias a este proyecto he podido adquirir conocimientos sobre nuevas tecnologías que antes ni tan solo conocía. También he desarrollado mis conocimientos y capacidades en sistemas Linux más allá de los conocimientos de uso básico que tenía al inicio de este proyecto.

De esta manera he profundizado en conocimientos que ya tenía como podría ser la gestión de un servidor DHCP, y administración básica. Pero he aprendido desde 0 cómo funciona un servidor tftp, un servidor NFS, cómo funciona el arranque y el inicio de un sistema operativo. También he podido profundizar en materias que si bien se explican durante la carrera, se hace de manera escueta y puntual, como son las dos tecnologías de virtualización utilizadas durante el proyecto.

Por último, el desarrollo tan accidentado de la primera parte del proyecto y la poca documentación disponible me ha hecho mejorar tanto a nivel de búsqueda de información, ya se encontrarla o diferenciar si era correcta o no, como a controlar la frustración al estar temporadas sin avanzar o avanzando muy poco.

Por esta razón pienso que este proyecto a sido enriquecedor pues por una parte he obtenido conocimientos y aptitudes técnicas que antes no disponía pero a su vez también he mejorado aptitudes personales que me ayudarán a desarrollar y gestionar otros proyectos en el futuro.

7.3 Trabajos futuros

Una vez acabado el proyecto, se pueden destacar un par de trabajos futuros tanto para la mejora tecnológica del proyecto y que por lo tanto mejoren aún más el funcionamiento de la empresa.

- **Migración** del resto de máquinas al clúster: hay que recordar que este proyecto se ha realizado sobre un hardware disponible en la empresa pero que no estaba siendo utilizado. Se podría migrar parte del hardware empleado al clúster de manera fácil para que este sea gestionado de forma centralizada.
- **Clúster híbrido**: si bien es cierto que la implementación del proyecto permite aumentar la eficiencia de uso de los recursos de la empresa, estos siguen siendo limitados y por lo tanto si la empresa crece esta se vería obligada o a comprar más hardware o a mover parte de su infraestructura a clouds privados como podrían ser **amazon web services** o **DigitalOcean** entre otros. Un cluster híbrido permitiría gestionar de manera centralizada tanto las máquinas propias de la empresa como las instancias contratadas a terceros.
- **Linux containers**: si bien la empresa ya utiliza linux containers dentro de las máquinas virtuales desplegadas por opennebula, se podría implementar alguna tecnología para gestionar containers de manera centralizada ya sea empleando docker machine con el driver de opennebula o empleando sistemas con **MesOS** o **RancherOS** para desplegar un cluster.

7.4 Impacto medioambiental

El uso más eficiente del hardware disponible, permite realizar y desplegar las mismas funciones y servicios que antes con menos hardware reduciendo así la cantidad de energía consumida por el centro de datos, ya sea la consumida por las máquinas como por el resto de la infraestructura como podría ser el sistema de refrigeración, y reduciendo la emisiones derivadas de su producción.

REFERENCIAS

[1] Gartner says efficient data center design can lead to 300 percent capacity growth in 60 percent less space (http://www.gartner.com/newsroom/id/1472714)

[2] Linux (http://www.linuxfoundation.com)

[3] KVM (https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine)

[4] Xen (http://wiki.xen.org/wiki/)

[5] Docker (https://www.docker.com/)

[6] OpenStack (https://www.openstack.org/)

[7] OpenNebula (http://opennebula.org/)

[8] Why raid 6 stops working in 2019 (http://www.zdnet.com/article/why-raid-6-stops-working-in-2019/)

[9] OpenNebula design and installation guide (http://docs.opennebula.org/pdf/4.14/opennebula_4.14_design_and_installation_guide.pdf)

[10] Khoa Huynh; Stefan Hajnoczi (2010). "KVM/QEMU Storage Stack Performance Discussion"

GLOSARIO DE TERMINOS

Kernel: software que constituye la parte principal y fundamental de un sistema operativo, se encarga de proporcionar al software acceso al hardware y gestionar los servicios entre otras funcionalidades.

Frontend: capa de software con la que interactúa un cliente

Vlan: las vlans son redes lógicas que funcionan de manera autónoma dentro de una misma red física

Cluster: es un conjunto de dos o más máquinas que se caracterizan por mantener una serie de servicios compartidos y por estar constantemente monitorizándose entre sí

Hipervisor: plataforma que permite aplicar técnicas de virtualización en una computadora.

RAID: conjunto redundante de discos que utiliza diversas unidades de almacenamiento para almacenar datos de manera distribuida y/o con replica de datos.