UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

# IMPLEMENTACIÓN DE UN BALANCEADOR DE CARGA Y ALTA DISPONIBILIDAD PARA SERVICIOS WEB

# T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

JORGE EDWARS VEGA ROMO

DIRECTOR DE TESIS: M. I. JORGE ARMANDO RODRÍGUEZ VERA

CIUDAD UNIVERSITARIA AGOSTO 2015

# AGRADECIMIENTOS

*A la Universidad Nacional Autónoma de México por permitirme haber estudiado y prestarme los elementos necesarios para llevar a cabo el aprendizaje que he tenido a lo largo de mi carrera.*

*A la Facultad de Ingeniería y a todos los profesores que me dejaron una gran enseñanza para alcanzar mis metas profesionales y darme buenas bases para ser un buen profesionista.*

# DEDICATORIAS

***A mi mamá:***

*Muchas gracias por todo el apoyo y la ayuda necesaria para llegar hasta este punto y alcanzar esta meta.*

***A Vianey mi amor de mi vida:***

*Muchas gracias por todo el apoyo, cariño y amor que me has brindado no sólo en la etapa de mi tesis, si no desde que estaba estudiando, ya que gracias a ti me diste un motivo para seguir adelante cuando ya no sabía que camino debía de seguir. Te amo mi amor.*

***A mi familia:***

*Por brindarme todo el apoyo y la compresión necesaria durante esta etapa de mi vida.*

***A mis amigos:***

*Gracias por todo el apoyo, amistad y grandes momentos que pasamos a lo largo de nuestra estadía en la Facultad y recuerden nuestros "Paseos Infinitos".*

CAPÍTULO III. IMPLEMENTACIÓN DE LA SOLUCIÓN

# INTRODUCCIÓN

Actualmente las páginas web han sido un gran recurso que tienen las personas para poder buscar información, contactarse con familiares, amigos e inclusive desconocidos, manejar cuentas bancarias, entre otras cosas. Lo cierto es que cuando inició la internet no se tenía esta ventaja, ya que pocas personas podían tener una computadora con dicho recurso; esta tecnología era casi reservada para instituciones gubernamentales y grandes universidades. Conforme la tecnología fue avanzando, tener un equipo de cómputo que tuviera acceso a internet fue más sencillo, con esto las páginas de internet empezaron a tener más problemas, no sólo en la cantidad de usuarios que cada día se sumaba a internet, si no que también empezaron a realizarse ataques a las diferentes páginas de internet, lo anterior para conseguir información de cualquier tipo o inhabilitar la página por un tiempo indeterminado. Esta problemática repercute directamente a los encargados de estas páginas, por eso se comenzaron a crear opciones para que se pudieran evitar bajas en éstas.

Por lo anterior, el siguiente trabajo de tesis desarrolla la Implementación de un Balanceador de Carga y Alta Disponibilidad para Servicios Web, el cual ayudará a minimizar la problemática de la caída de los servicios de una de las páginas más solicitadas en la Facultad de Medicina Veterinaria y Zootecnia, ya que esta página ayuda a los alumnos y profesores a interactuar entre ellos para mejorar la experiencia de aprendizaje.

En el primer capítulo de este trabajo, se presenta un marco teórico que nos ayudará a comprender todos los conceptos que se manejarán en el capítulo 2 y 3, así como una breve explicación de lo que hace un balanceador de carga y la alta disponibilidad.

En el segundo capítulo, se observa la problemática que se tiene en la Facultad, en donde nace esta necesidad y se ve la solución que se necesita para corregir esta problemática.

En el tercer capítulo, se muestra la implementación de la solución propuesta del capítulo 2, así como también la migración de los servicios web a la nueva estructura que se necesita para que la solución funcione de manera correcta, al final de este capítulo se hacen ciertas consideraciones que deberán ser leídas antes de replicar cualquier punto de este trabajo.

Finalmente se muestran las conclusiones del trabajo y la bibliografía consultada.

# CAPÍTULO I

# MARCO TEÓRICO

En la actualidad las páginas de internet son visitadas por millones de personas, las cuales buscan ayudar con algo en específico, ya sea consultar información, entrar a alguna red social, páginas educativas o realizar transacciones de sus cuentas bancarias. Para esto, las personas necesitan que la página a visitar, se encuentre siempre activa, pero esta necesidad no es tan sencilla de cubrir por la gente que trabaja en el área de las tecnologías de la información, ya que aún teniendo una gran cantidad de recursos, hemos visto a páginas como google, gubernamentales, entre otras que se quedan fuera de línea, por eso en el capítulo siguiente se presentará una solución a este problema que si bien no está exento de algún ataque informático, se tratará de evitar que la página de internet deje de funcionar.

## 1.1 ALTA DISPONIBILIDAD

Se puede definir a la alta disponibilidad como la capacidad de un sistema para mantener activo un determinado servicio[1] ó en su defecto la recuperación del mismo en un tiempo tan pequeño que el usuario no note el fallo del sistema, esto lo realizará haciendo que otra máquina tome el rol principal de las actividades para tener un sistema continuo sin afectar al negocio.

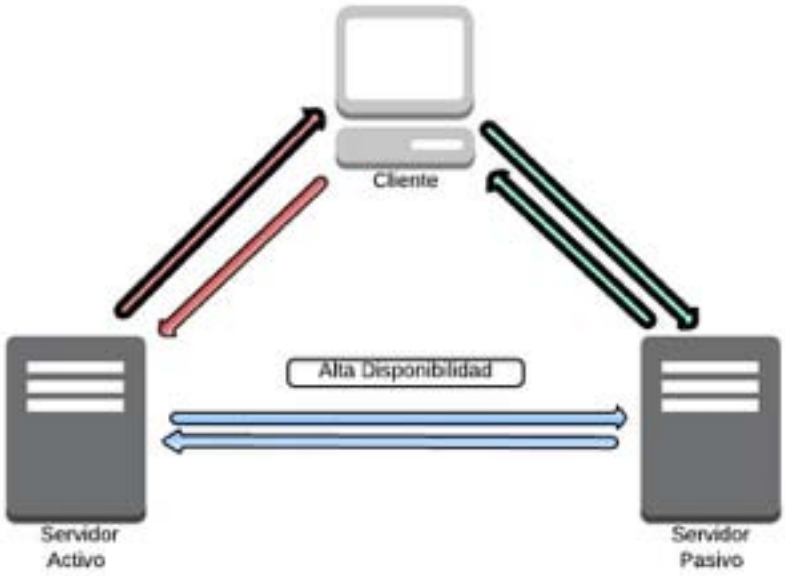



Figura 1.1. Representación de un modelo de alta disponibilidad

En la figura 1.1 podemos ver una representación de cómo se vería la alta disponibilidad, donde nuestro cliente tratará de realizar una conexión con el servidor activo representado por medio de la flecha en color rojo con bordes claros, si el servidor activo se encuentra en funcionamiento responderá la solicitud que se representa por medio de la flecha roja con bordes obscuros, el servidor activo estará en constante comunicación con el servidor pasivo representado por medio de las

[1] Servicio en el área de las tecnologías de la información se refiere a un conjunto de actividades que buscan atender las necesidades de un cliente.

flechas azules, en dado caso que el servidor activo no esté funcionando y el cliente quisiera realizar una conexión con el servidor activo, el servidor pasivo tomaría la solicitud y se haría la conexión con el cliente (en este caso se representan con las flechas verdes con bordes obscuros), este proceso será transparente para el cliente y su conexión no se verá afectada.

La alta disponibilidad se puede dividir en dos tipos:

Por *hardware*:

> Se le conoce también por redundancia de *hardware,* esto se refiere que en dado caso que se produzca cierto fallo en el *hardware* (puede ser en la energía eléctrica, problemas de red, equipos de red, etc.), si se tiene esta redundancia podemos cambiar el *hardware* en *hotswap*[2] sin tener que sufrir la baja de los servicios de la organización.

Por aplicaciones:

> En dado caso de que se produzca un fallo en las aplicaciones (servicios), con un *software* de alta disponibilidad podemos volver a levantar el servicio sin la necesidad de hacerlo manualmente, los servicios que hayan fallado en alguno de los nodos del *cluster*[3]. Cuando un nodo que ha fallado recupera los servicios se tienen dos opciones:

[2] Capacidad de algunos componentes esenciales que permiten instalarse o sustituirse sin detener las operaciones normales del equipo de cómputo donde se encuentra.

[3] El significado de *cluster* y nodo se verán en el punto 1.1.1

- La primera es cuando el nodo principal presenta un fallo y vuelve a tener activo el servicio, el nodo vuelve a ser de nuevo el nodo principal.
- La segunda es cuando el nodo principal presenta un fallo y vuelve a tener funcionamiento, el nodo estará al pendiente del nodo activo (secundario) ya que si este falla el primer nodo volverá a funcionar como el principal.

La alta disponibilidad nos permite tener activos nuestros servicios casi un 100% del tiempo, esto es un reto bastante difícil de realizar para las necesidades del negocio, ya que repercute directamente en los costos que podría generar si estuviera apagado y por supuesto con la confiabilidad que tendrá el usuario al saber que el servicio (en algunos casos su información) está siempre disponible cuando se le necesite.

Características:

Se debe de contar con lo siguiente para la generación de alta disponibilidad:

- *Cluster:* servidores duplicados N+1 para una falla de alguna máquina.
- Balanceo de carga: Distribución del tráfico entrante entre distintos servidores para soportar y distribución de las peticiones.
- Escalabilidad: Capacidad de crecimiento a la medida que aumenta la demanda de trabajo pero siempre brindando un nivel de rendimiento aceptable.

### 1.1.1 *Cluster*

Se define a un *cluster* como un conjunto de máquinas que realizan la misma tarea, un cluster está construido mediante *hardware* con la finalidad de comportarse como una sola máquina, las máquinas toman el nombre de “nodos”.

Clasificación de los c*lusters:*

La clasificación de los *clusters* depende del uso que se les vaya a dar y de los servicios que vayan a ofrecer, una vez sabiendo esto los podemos clasificar de la siguiente manera:

- Alto rendimiento: Este tipo de *clusters* se utilizan para cuando se necesiten ejecutar tareas que requieran una gran capacidad computacional, gran cantidad de memoria o ambas al mismo tiempo. La desventaja de este tipo de *cluster* es que al llevar a cabo las tareas requeridas el sistema se puede comprometer por periodos muy grandes de tiempo.
- Alta disponibilidad: La principal función de estos servidores es proveer disponibilidad y confiabilidad de un cierto servicio. Para ofrecer esto se necesita un *software* que detecte fallos y permita recuperarse del mismo y en el *hardware* se evita tener un punto único de fallo.
- Alta eficiencia: Tienen como objetivo realizar la mayor cantidad de tareas en el menor tiempo posible.

Existen 2 tipos de *clusters*, esto es dependiendo de la estructura de los nodos:

- Activo/pasivo: En este tipo de *cluster* hay al menos un nodo sin operación a la espera de algún fallo, cuando el fallo suceda el nodo que se encuentra sin operación pasará a ser el que responda las solicitudes de los clientes.
- Activo/activo: En este tipo de *cluster* todos los nodos están activos y todos brindan un servicio y se estarán respaldando entre sí.

Los servicios que pueden incluirse en un *cluster* son los siguientes:

- Servidores de bases de datos.
- Servidores web.
- Servidores de balanceo de carga.
- Servidores de correo.
- Servidores *firewall*[4].
- Servidores *DNS*[5].
- Servidores *DHCP*[6].
- Servidores *proxy cache*[7].

[4] *Firewall*: Es un dispositivo que funciona como un cortafuegos entre redes, permitiendo o denegado las transmisiones de una red a la otra.

[5] *Domain Name System*: Sistema de nombres de dominio

[6] *DHCP*: Es un protocolo que permite a dispositivos individuales obtener su propia información de configuración de red a partir de un configuración automática.

[7] *Proxy Cache*: Permite incrementar la velocidad de acceso a internet al mantener localmente las páginas más consultadas por los usuarios de una organización, evitando las conexiones directas con los servidores remotos.

### 1.1.2 *Mirror*

Se puede definir al *mirror* como un servidor que será la copia exacta de algún otro. Este tipo de servidores sirven para disminuir el tiempo de acceso del usuario a servidores que están situados en diferentes lugares, usualmente muy retirados entre sí.

## 1.2 Balanceo de carga

El termino “balanceo de carga” se refiere a una división de la carga del trabajo que tiene algún sistema donde esa división se realizará en todos los nodos del *cluster* que se tengan disponibles.

En otras palabras, un balanceador se encarga de recibir todas las peticiones de un cierto servicio que se presta en el *cluster;* una vez que las recibe, se encarga de volver a reenviar a los servidores dentro del *cluster* donde serán atendidas. Esto sucede de forma tan transparente que los usuarios que solicitan el servicio no notan que entran a este proceso, ya que el usuario no tiene el conocimiento de que existe un balanceador de carga.

Existen diversas formas de realizar un balanceador de carga, por *hardware* , por *software* o una combinación de los dos.

El balanceador de carga es ideal cuando es muy difícil pronosticar qué tan grande podrá ser la carga de trabajo que se va a tener para los servicios que se prestarán.

### 1.2.1 Balanceadores por *hardware*

La manera más simple de balancear la carga de trabajo hacia un determinado servicio es utilizar un dispositivo de *hardware* que lo implemente. Aunque tienen una gran potencia, estabilidad y escalabilidad, hay varios factores que convierten al balanceo por *hardware* en factor de duda para su realización; uno de ellos es la configuración y el mantenimiento del dispositivo que en algunos casos puede llegar a ser muy problemático, pero el verdadero problema radica en el precio, esto es muy importante mencionarlo porque muchas organizaciones no cuentan con los recursos necesarios para comprar este tipo de dispositivos.

En la figura 1.2, se puede apreciar el modelo de un balanceador de carga por *hardware*, se observa que el cliente hace una conexión de internet hacia un servidor, pero el que toma la solicitud es el balanceador, dependiendo de la configuración de éste, mandará la solicitud a un servidor determinado y éste responderá la solicitud del cliente; si más de un cliente tratara de hacer conexión, el balanceador podrá dirigir el trabajo a otros servidores o seguirá mandando el trabajo al servidor que ya tiene la conexión.

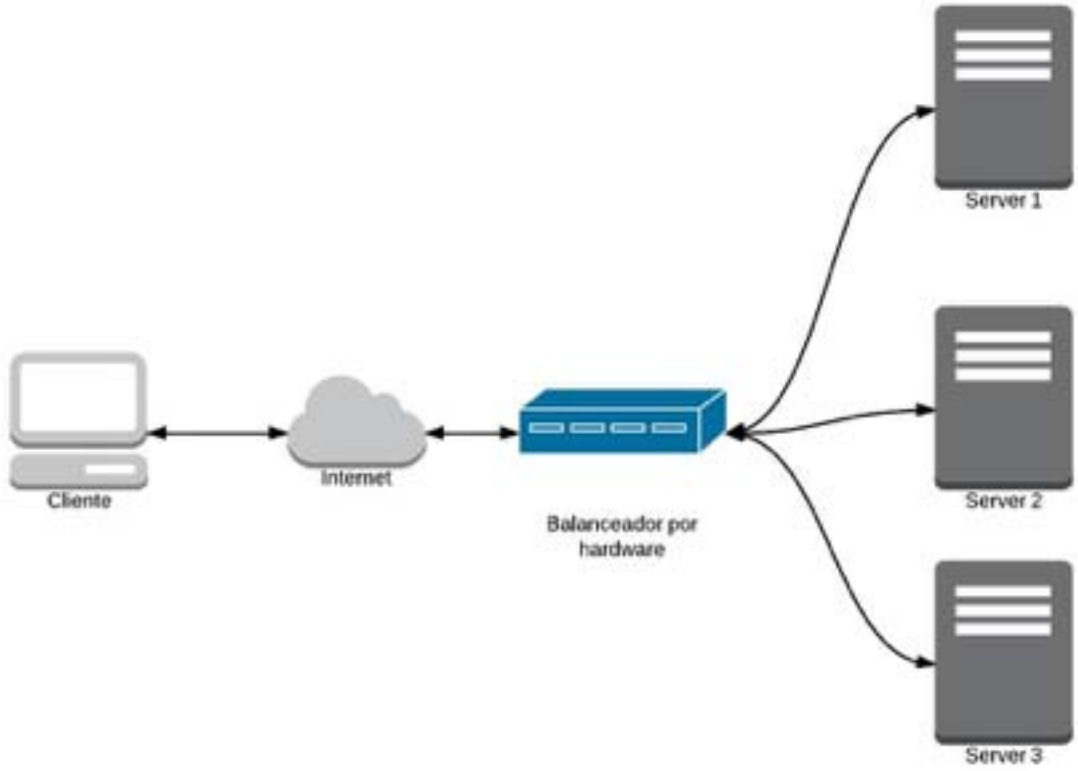



Figura 1.2 Modelo de un balanceador de carga por *hardware*

### 1.2.2 Balanceadores por *software*

El balanceo de carga por *software* es una forma de balancear la cantidad de tráfico entrante a un cierto servicio a través de un programa que está adentro del sistema operativo o como una aplicación complementaria. Este tipo de solución suele ofrecer facilidad para realizar tanto la implementación como el mantenimiento y tiene un rendimiento muy similar a un balanceador de carga basado en *hardware*.

En cuanto a las ventajas de este tipo de balanceadores está el precio, porque si bien la mayoría de los balanceadores son *software* libre también hay balanceadores que necesitan una cierta licencia, sin embargo el precio de un balanceador por *software* es mucho menor que el de un físico. Otra ventaja muy importante es cuando se necesite un servidor de mayor potencia para utilizar como balanceador, el anterior lo podemos reciclar y podemos ocuparlo en alguna otra tarea.

Las desventajas más claras es que se necesita una mayor cantidad de tiempo para su mantenimiento y que se tiene una menor potencia comparada con la que nos podrían ofrecer los balanceadores por *hardware.*

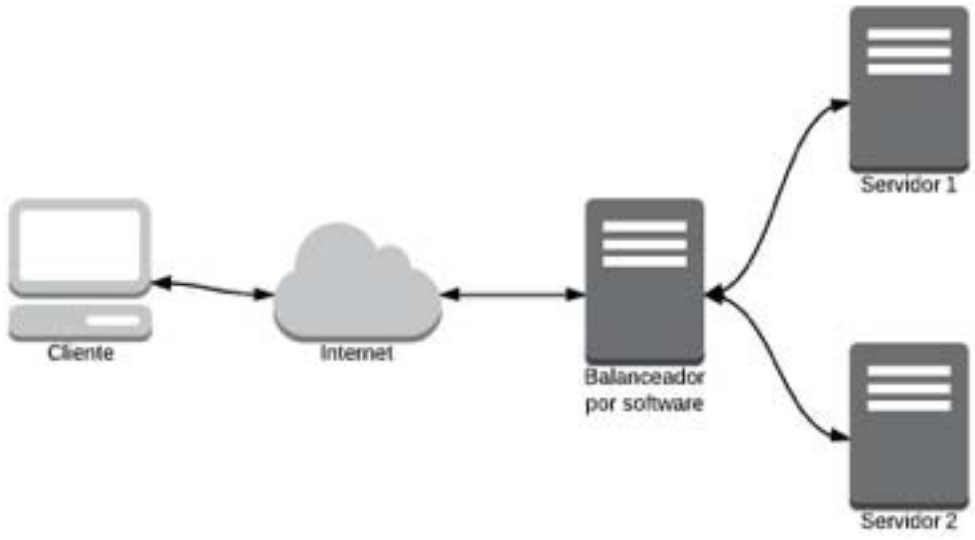



Figura 1.3 Modelo de un balanceador de carga por *software*

En la figura 1.3, vemos la representación de un balanceador de carga por *software*, donde nuestro cliente tratará de hacer una conexión con nuestro servidor. Aquí el balanceador de *software* se representa como un servidor, pero en realidad su única función será mantener el servicio de balanceo, lo que realizará es que mediante la configuración que éste mantenga, enviará la petición de conexión al servidor que esté especificado en su configuración, si más de un cliente intentara hacer conexión sucederá de la misma manera que con el balanceador por *hardware*, dependiendo de su configuración mandará la solicitud al servidor que ya tiene conexión o al que esté libre de ellas.

Antes de hablar de una de las características de los balanceadores de carga por *software* tenemos que hablar del modelo OSI[8], ya que nuestro balanceador por *software* ocupa este modelo como ayuda. En la figura 1.4, vemos el modelo OSI, el cual se compone de 7 capas cada una hace una función diferente. A continuación se hablará un poco de ellas:

1. La capa física: Esta capa se encargará de la transferencia de bits por medio de un canal de comunicación. La capa física debe tener en cuenta que si manda un bit de origen, debe de recibir otra máquina el bit de destino.
2. La capa de enlace: La principal tarea de esta capa es la corrección de errores. También se encarga de separar la entrada de datos en tramas[9], las cuales envía y procesa devolviéndolas al receptor.

[8] Modelo propuesto para estandarizar la interconexión de sistemas abiertos, este modelo establece los lineamientos para que el software y los dispositivos de diferentes fabricantes trabajen juntos.

[9] trama es una serie sucesiva de bits, organizados en forma cíclica y que es una forma de transmitir información.

3. La capa de red: Tiene como función el control de la operación de la subred. Esta capa debe determinar la ruta que tiene que trazar los paquetes de origen al destino y en algunos casos determina diferentes soluciones. También tiene como problema la congestión de la red, así como la responsabilidad de resolver problemas de interconexión entre redes con diferentes protocolos.
4. La capa de transporte: La principal función de esta capa consiste en aceptar los datos de la capa de sesión, dividirlos en partes más pequeñas, pasárselos a la capa de red, para así asegurar que lleguen al otro extremo de la comunicación sin problema alguno de la manera más eficiente. Esta capa a diferencia de las otras ya antes mencionadas, se le considera de origen-destino, en otras palabras que la máquina que inicia origen y destino compartan el mismo programa para que se realice correctamente la comunicación.
5. La capa de sesión: Esta capa se encarga de que usuarios con máquina diferentes puedan establecer una sesión (conexión). Esta sesión puede permitir a los usuarios acceder a un sistema compartido temporalmente que no esté cerca del usuario, o bien transferir archivos de un usuario a otro.
6. La capa de presentación: Su función es el ordenamiento que se transmite, a esta capa no le interesa el movimiento de los bits de origen a destino. Tiene como función adicional la codificación de datos.
7. La capa de aplicación: Esta capa es en la que los usuarios finales realmente verán la información. Esto lo hace con editores y programas donde el usuario visualizará la información y hasta podrá editarla.

| |
|---|
| 7.- Aplicación |
| 6.- Presentación |
| 5.- Sesión |
| 4.- Transporte |
| 3.- Red |
| 2.- Enlace |
| 1.- Física |

Figura 1.4 Modelo OSI

Con lo antes visto ahora podemos hablar acerca del funcionamiento de los balanceadores de carga por *software* a nivel de red. Tradicionalmente los balanceadores de carga por *software* trabajan en la capa 4 del modelo OSI, donde se establece una conexión y es balanceada hacia algún servidor que se encuentre disponible en ese momento. La forma más simple de usar el balanceo de carga a través de la capa 4 es por medio de un algoritmo *roundrobin*[10]. Aquí se toma cada conexión entrante y la dirigimos al primer servidor listado en el *backend*[11], seguirá haciendo lo mismo hasta que cada una de las peticiones sean enviadas a cada *backend* de los servidores. Para esto sólo se necesita que en el balanceador de carga exista una lista de los servidores y unas variables que marquen el último servidor usado. Pero últimamente los balanceadores de carga han empezado a trabajar con la capa 7 del

[10] Este algoritmo consiste en formar las peticiones en una fila y como vayan llegando van a ser mandadas a los nodos no importando de donde vengan ni tampoco la capacidad de los nodos.

[11] Hace referencia al estado final de un proceso.

modelo OSI, donde en vez de revisar las cabeceras ip[12] de la capa de transporte para elegir el balanceo, los balanceadores de la capa de aplicación lo realizarán a través de peticiones de HTTP[13] evitando así ataques a través de las cabeceras con algún contenido malicioso. La ventaja es que nos permite mirar la petición y sus encabezados y a partir de esto podemos hacer nuestra estrategia de balanceo. Algunas formas de usar esta información es para balancear peticiones basandonos en la consulta, con *cookies*[14] *ó* algún encabezado que nosotros escojamos el cual puede ser información de la capa 4 (incluyendo dirección ip origen del solicitante y dirección ip destino del servidor), así como en la capa 4 la forma más fácil de balancear es a través de un algoritmo *roundrobin*, en la capa 7 es mejor por la *URL*[15], con esto podemos asegurar que todas las peticiones por un recurso específico van a ser dirigidas a un servidor en particular.

## 1.3 Servicios Web

La W3C[16] define servicio web como un sistema de *software* diseñado para permitir la interoperatibilidad máquina a máquina en una red. En otras palabras un servicio web puede ser cualquier sistema de *software* diseñado para soportar interacción entre un cliente y un servidor a través de la red.

---

[12] Estas cabeceras contienen ip origen, ip destino, versión, tamaño, tipo de servicio a requerir, longitud total, identificador, banderas de estado, numero de fragmento del paquete, tiempo de vida, protocolo, opciones y un relleno.

[13] *Hiper text Transfer Protocol* (protocolo de transferencia de hipertexto) es el método más utilizado para el intercambio de información a través de la red y por el cual se transfieren las páginas web a una computadora o dispositivo.

[14] Se refiere a un archivo de texto pequeño que los sitios web almacenan en la computadora.

[15] Localizador de recursos uniforme (*Uniform Resource Locator*) se utiliza para nombrar recursos de internet que sirve para su localización o identificación.

[16] World Wide Web Consortium (W3C), es un consorcio internacional que produce recomendaciones para internet.

Algunas ventajas de los servicios web son:

- Aumenta la interoperatibilidad entre programas independientemente de la plataforma en donde están instalados.
- Fomentan los estándares y protocolos basados en texto, haciendo más fácil acceder y entender su contenido y funcionamiento.
- Al utilizar el protocolo HTTP, puede utilizar un sistema de seguridad sin cambiar las reglas de filtrado del sistema.

Dentro de los servicios web también existe un balanceador de carga, pero éste es especialmente para un servidor web *apache*[17], que dentro de la estructura del servicio se incluyen varios módulos que pueden soportar el balanceo de carga como son *lbmehtod* y *mod_proxy_balancer*[18]*;* estos módulos se tienen que haber cargado con anterioridad para su utilización. Actualmente podemos encontrar que hay tres algoritmos disponibles para realizar el balanceo de carga por medio de *apache* los cuales son:

- Conteo de solicitud
- Tráfico ponderado de conteo
- La espera de contar solicitudes

[17] Servidor HTTP

[18] Los cuales no se verán en este trabajo.

## 1.5 Pruebas a realizar

Para cualquier implementación es fundamental la parte de las pruebas, esto nos ayudará a verificar si nuestra configuración está funcionando adecuadamente y principalmente que lo que nosotros queremos de estos servicios sea lo que nos estén entregando.

Dentro del *cluster* de alta disponibilidad con balanceo de carga es fundamental ver que las dos partes que lo conforman estén funcionando adecuadamente, por esa razón se necesitan hacer pruebas para ver que su funcionamiento sea óptimo. Dentro de la alta disponibilidad haremos pruebas de conexión entre nuestros nodos y para el balanceo tendremos una prueba donde veremos que realmente distribuya la carga de las peticiones que nos solicitarán.

### 1.5.1 Prueba de conexión

Las pruebas a realizar dentro de la alta disponibilidad en sí son muy pocas, pero con esto nos garantizará su buen funcionamiento, una vez que sabemos que la alta disponibilidad tiene como función mantener los servicios activos, la prueba básica es exactamente ver que nuestro servicio esté activo aun si algún nodo del *cluster* falla por algún motivo (problemas de red, o de energía, etc.), para esto una vez que tenemos funcionando nuestro *software* o *hardware* de alta disponibilidad lo que procederemos a realizar es que intencionalmente apagaremos el nodo activo (solo en el caso de *cluster* del tipo activo/pasivo) y verificaremos de manera remota si el servicio que mantiene ese nodo sigue activo. Si la alta disponibilidad fue bien configurada y está activa no debería de tardar más de 5 segundos en tener activo el servicio pero claro en otro nodo del cluster, si el servicio permanece activo nuestra alta disponibilidad está bien realizada. Después procederemos a volver a prender el otro nodo (nodo principal), aquí lo que podemos observar es que después de

prenderlo existe algún problema entre los nodos, ya que suele suceder que al poner mal la configuración para saber qué nodo es el que mantendrá el servicio si el que lo tiene actualmente o el nodo principal, si al analizar que sólo un nodo tiene el servicio en funcionamiento, si esto sucede sin ningún problema podemos ver que nuestro servicio de alta disponibilidad está funcionando correctamente ya que no sólo podemos ver que nuestro servicio está funcionando a pesar de que el nodo principal está caído, también podemos darnos cuenta de que nuestros nodos están teniendo la comunicación necesaria en caso de que alguno de ellos fallara repentinamente.

En el caso de un servidor de alta disponibilidad activo/activo, lo único que debemos verificar es que el servicio esté activo en ambos nodos y lo más importante es si nuestros nodos están disponibles, porque cuando se realice el balanceo sabremos qué nodo está listo para balancear y cual no. Para verificar que se estén comunicando debemos de entrar a los *logs*[19] del servicio de balanceo (sólo en el caso de alta disponibilidad por *software*), verificamos al final del archivo que ambos nodos se estén comunicando y sepan que el otro nodo está activo, para la alta disponibilidad por *hardware* debemos de entrar al dispositivo y verificar que nuestro dispositivo de alta disponibilidad tenga ambos nodos activos y por supuesto que estén en constante comunicación.

### 1.5.2 Prueba de balanceo

Dentro del balanceador de carga las pruebas son muy sencillas, después de la configuración ya sea del *software* o *hardware* que nos ayudará a balancear tenemos que probar que realmente lo haga, pero aquí hay un problema, si ponemos ya en funcionamiento nuestro servicio web que balancearemos no podremos notar si en realidad se está balanceando, para hacer la verificación de que el balanceo funciona, nos dirigiremos a la configuración de nuestro servicio web, una vez ahí dentro le

[19] Son los archivos que dan reportes acerca del estado del sistema o algún servicio.

diremos que acepte pocas conexiones, después nos vamos a crear un página web de prueba con cualquier nombre, repetimos el mismo proceso en los demás nodos que estarán dentro de nuestro *cluster* de balanceo de carga pero con la única excepción de que la página de prueba debe de tener un nombre distinto por cada uno de los nodos. Después de hacer estos cambios mandaremos la solicitud del servicio a través de nuestro navegador web y veremos alguna página de prueba (depende mucho de nuestro algoritmo de balanceo), lo que haremos a continuación es mandar muchas peticiones y podremos ver el funcionamiento del balanceo, una vez que pudimos ver que nuestro servicio está balanceado, ponemos nuestra página principal de nuestro servicio web con la configuración que nosotros propongamos para el número de conexiones permitidas, ya que ahora sabemos que nuestro balanceador está funcionando adecuadamente.

## 1.6 Errores a visualizar

Como en la mayoría de las implementaciones siempre habrá errores que podemos llegar a ver, en el caso de un *cluster* de alta disponibilidad con balanceo de carga los errores más comunes son:

- Por parte de la alta disponibilidad es que no haya una buena comunicación entre nuestros nodos, con esto se refiere a que ninguno de los nodos que estén dentro nuestro *cluster* tenga el conocimiento de que existen otros nodos o si existen pero no están en funcionamiento, este error es más común llegar a visualizarlo dentro de un servidor de alta disponibilidad del tipo activo/pasivo ya que el servicio no se iniciará y ambos nodos tendrán dentro de sus *logs* que se están tratando de comunicar pero ninguno de los dos llega a responder, este error es por parte de una mala configuración y en este caso se recomienda volver al archivo de configuración y verificar que nuestra configuración sea la idónea para que este error desaparezca, en el caso de activo/activo es

recomendable estar frecuentemente revisando los *logs* para verificar que todo esté en orden.

- En el caso del balanceo de carga lo que podemos llegar a notar como error es que alguno de los nodos no esté balanceando (esto claro se vera durante la etapa de pruebas), aquí puede llegar a ser por una mala configuración de nuestro algoritmo, ya que si no lo configuramos de una manera adecuada podemos dejar a nuestro servidor más potente como el último en ser tomado. Para arreglar esto sólo debemos de cambiar la configuración de nuestro algoritmo, otro posible error que llegamos a ver es que no haga el balanceo, este error es usual que aparezca si nuestro balanceador no se está comunicando con nuestros nodos (la forma en que lo hace depende del balanceador), tendremos que verificar primero nuestra configuración, si notamos que ahí no hay ningún problema nos vamos a nuestros nodos a ver si cumplen con los requisitos del balanceador para la comunicación.

Resumen del capítulo:

Al final del capítulo pudimos definir y entender la utilidad de un balanceador de carga y de la alta disponibilidad, así como también llegamos a definir su estructura, funcionamiento y las clasificaciones que tienen. Pudimos ver que la alta disponibilidad tiene como función mantener nuestros servicios activos, esto lo hace por medio de un servidor de respaldo o que puede ser un mirror, también vimos que un balanceador de carga se encarga de distribuir la cantidad de solicitudes de un servicio entre dos o más servidores y por supuesto vimos algunos errores comunes a la hora de tratar de implementar un servidor de balanceo de carga con alta disponibilidad.

# CAPÍTULO II

# PLANTEAMIENTO DEL PROBLEMA Y PROPUESTA DE LA SOLUCIÓN

Dentro de la Facultad de Veterinaria Medicina y Zootecnia de la Universidad Nacional Autónoma de México hay varias páginas de internet que exclusivamente son de uso interno, pero hay algunas como “Apuntes en línea” que están disponibles para todo el público a nivel mundial. En dicha página existen cursos para alumnos de la facultad donde pueden subir sus tarea, hacer exámenes, etc. Debido a la utilidad que tiene la página es indispensable que siempre esté activa para que los usuarios puedan ver su información cuando ellos la necesiten.

## 2.1 PROBLEMÁTICA ACTUAL

La página “Apuntes en línea” es una de la páginas más utilizadas de la FMVZ, siendo ésta un servidor moodle 2.5 que contiene videoclases, audiolibros, memorias electrónicas y recursos adicionales para cursos presenciales. Es un proyecto PAPIME PE 203311, del profesor Germán Valero, donde no sólo los alumnos pueden tener acceso a ésta, sino que además personas de todo el mundo pueden ver parte de su contenido, debido a su gran utilización hay momentos en los que se han tenido algunos fallos en cuanto al acceso, tiempo de respuesta y visualización de la página. Por este tipo de problemas el Profesor Germán Valero decidió que se implementara un balanceador de carga y alta disponibilidad, se le encargó al departamento de servidores el desarrollo de la instalación y pruebas de la página ya antes mencionada, la tarea de realizar el sistema del Profesor se quedó a mi cargo para que hiciera la investigación pertinente, así como de la instalación y la documentación del mismo.

## 2.2 REQUERIMIENTOS A OBTENER

Dentro de las especificaciones que se nos solicitaron fueron las siguientes:

- Obtener los recursos de la máquina actual.
- Averiguar la máxima capacidad de conexiones que la página puede llegar a soportar.
- Instalar el balanceador de carga y alta disponibilidad.
- Probar la máxima capacidad de conexiones permitidas.

Ya obtenidos estos requerimientos lo siguiente fue buscar los recursos que tiene el servidor de la página de “Apuntes en línea” los cuales son:

- Servidor virtualizado.
- Memoria RAM de 9 GB
- CPUs 4
- Disco Duro 1 con 100 GB
- Disco Duro 2 con 20 GB
- Moodle versión 2.5.
- Sistema operativo CentOS versión 6.3.

El servidor actual cuenta con 4000 usuarios de los cuales utilizan esta plataforma para subir información requerida por los profesores, consultar información de la materia y realizar exámenes que requieren una gran cantidad de procesamiento y de respuesta, ya que para estos exámenes es fundamental contar con imágenes de alta calidad para que los alumnos tengan un mejor entendimiento del tema, por este motivo es importante que siempre se encuentre activo y en completa disposición para ser utilizado por los usuarios.

Ahora que hemos visto la capacidad del servidor de “Apuntes en línea” veremos cuántas conexiones soporta el servidor y buscaremos entre los archivos de configuración del *apache* hasta encontrar la siguiente palabra: *MaxKeepAliveRequest*[20]. Lo siguiente a realizar es ver el rendimiento del servidor con el número máximo de conexiones, para eso utilizaremos la herramienta de *apache* llamada ab, con lo cual vamos a poder ver los tiempos de respuesta de los usuarios; esto lo haremos mandando el máximo número de conexiones al servidor y

[20]Significa número máximo de conexiones permitidas.

veremos los resultados, el paquete de ab usualmente viene con el *apache* pero en dado caso puede instalarse de la siguiente manera:

yum -y install httpd-tools (debe de ser ejecutado con permisos de administrador)

Una vez que se instaló el paquete ejecutará lo siguiente:

ab -n 500 -c 100 http://apuntesenlínea.fmvz.unam.mx/

Donde el comando ab es el que se instaló a través del httpd-tools, la opción -n se refiere a la cantidad de conexiones que vayan a mandar para ver los tiempos de reacción del servidor y la opción -c se refiere a cuántas conexiones se harán concurrentes, las cuales no deben superar al número de conexiones totales, esta prueba se puede incrementar en el número de conexiones y conexiones concurrentes, dependiendo de nuestro servidor, para este caso nuestro servidor con estas 500 conexiones se empezó a alentar, llegamos al caso de obtener 1000 conexiones pero después de eso la base de datos no pudo resistir y falló, por lo que se tuvo que reiniciar el servidor. Por eso se recomienda que el número de conexiones no sea muy grande si se tiene una base de datos anidada al servicio web.
En la figura 2.1 veremos los resultados del comando ab, donde después de darnos la información del servicio viene cuántas conexiones se mandaron exitosamente, las cuales son las 500 que pusimos en nuestro comando, nos indica la versión de apache que utiliza nuestro servidor, su nombre y por cuál puerto se hacen las solicitudes, después el directorio raíz de nuestro sistema y su peso en *bytes*, el siguiente campo nos muestra las conexiones que se realizaron de modo concurrente y el tiempo que tomó en hacer todas las conexiones en segundos, después viene el número total de conexiones que se mandaron y cuántas fallaron, en este caso se ve que fallaron 171 de las 500 que se mandaron. Lo que sigue no tiene mucha relevancia hasta la sección que dice solicitudes por segundo, con esto se verá cuánto tarda nuestro servidor en

responder a una solicitud, en nuestro caso es 4.37 conexiones por segundo y también nos muestra cuánto tarda la conexión en segundos que son 22.9, además del tiempo cuando son concurrentes 0.23 también en mili segundos. Nos vamos a la parte final de lo que nos arrojó el comando ab, en esta parte vamos a ver el porcentaje de conexiones que se hicieron en un tiempo determinado que también está en mili segundos.

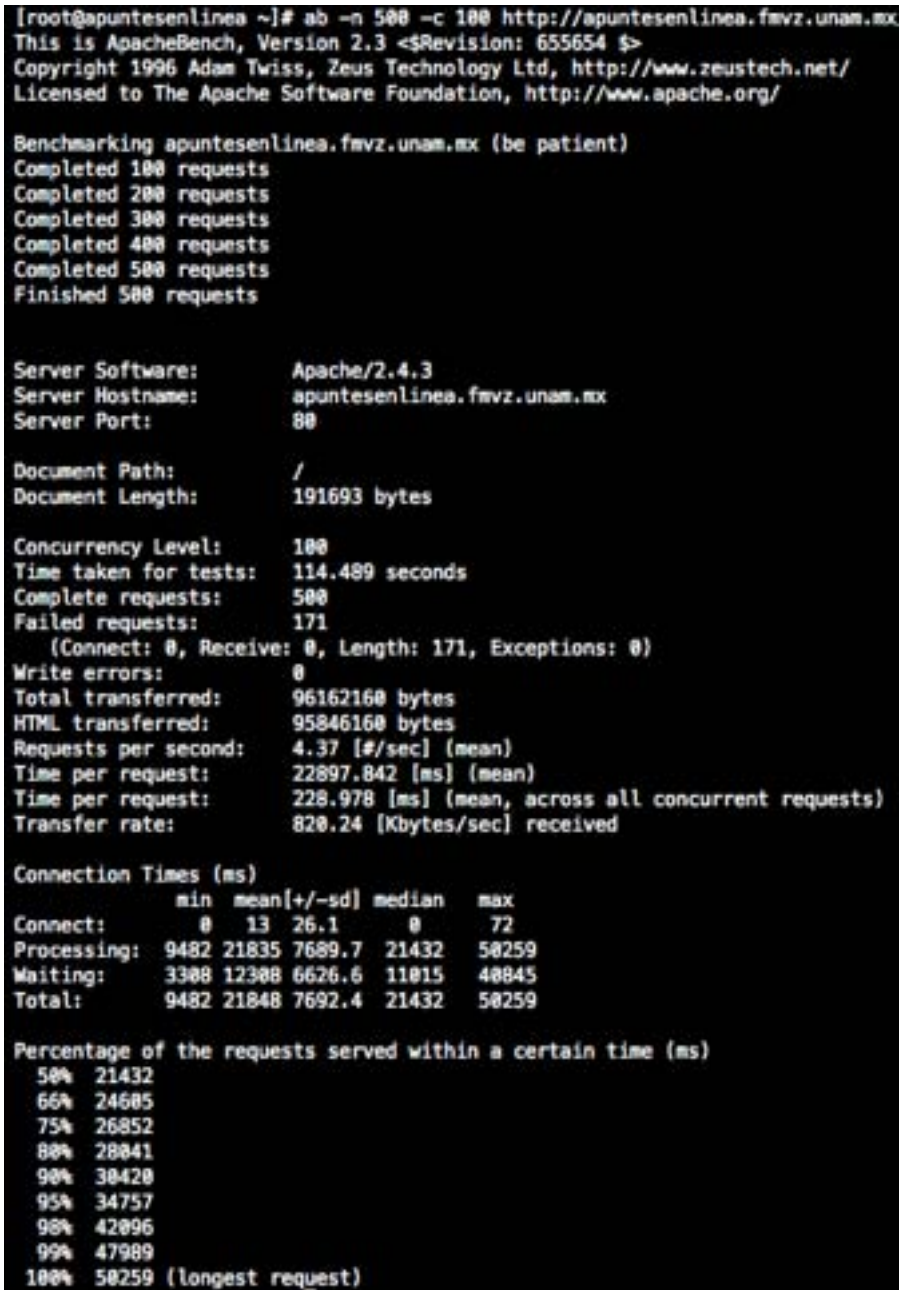



Figura 2.1 Resultados del comando ab

Con estos resultados lo que nos pide es que se optimicen los tiempos, de tal forma que no lleguen a los segundos; el tiempo de respuesta debe ser de la orden de mili segundos o que se tengan los mismos tiempos pero con más conexiones sin que el servidor tenga problemas, ni con la parte de la página web, ni con la base de datos, también que se pueda optimizar la cantidad de recursos en lo más posible.

## 2.3 RECURSOS FÍSICOS Y LÓGICOS

Nuestro sistema completo se encuentra virtualizado[21], por lo que los recursos físicos son más grandes de los que cuenta el servidor “Apuntes en línea”, por lo tanto, se podrían dar más recursos al servidor, aunque eso no nos ayudaría con nuestra meta de optimizar, la manera más óptima sería crear un balanceador de carga con alta disponibilidad, repartiendo los recursos del servidor de “Apuntes en línea”, así haremos más eficiente el servicio sin gastar tantos recursos. Aquí empieza el gran problema, ya que se decidió la manera de optimizar; para ello se necesita ver cómo se implementará el servicio de alta disponibilidad y el balanceo de carga, lo más óptimo sería comprar un dispositivo especializado para estas tareas, pero debido al bajo presupuesto que se tiene y a los altos precios de estos dispositivos, se ha optado por hacer el balanceo y la alta disponibilidad a través de programas especializados, pero la pregunta sería cuál es el mejor, por lo que se tienen que ver las opciones y decidir el mejor para esta ocasión.

[21] Se le llama virtualizado a un sistema que no se encuentra físico, si no más bien es creado por un software especial que crea recursos aislados.

## 2.4 BÚSQUEDA DE INFORMACIÓN PARA LA SOLUCIONAR LA PROBLEMÁTICA

Empecemos con los servicios de alta disponibilidad, que como sucederá con el balanceador de carga por la falta de presupuesto se buscará una opción de *software* libre, las soluciones de alta disponibilidad son las siguientes:

- *Heartbeat*
- *Ldirectord*

Estas dos opciones son las que se tiene más documentación para su instalación y su configuración, a continuación explicaré un poco más acerca de ellas.

*Heartbeat*

Es un servicio que provee alta disponibilidad a través de lo que el programa llama “latido de corazón”[22], lo cual a grandes rasgos es lo que hace. En realidad no manda un “latido”, más bien lo que sucede es que entre los servidores que están teniendo la alta disponibilidad se manda una notificación a través de la red cada cierto tiempo[23] para verificar si el otro o los otros servidores siguen activos, aquí depende mucho del tipo de *cluster* de alta disponibilidad que se tenga, si el “latido” falla en hacer conexión en un tiempo delimitado, lo que hará el servicio dependiendo del tipo de *cluster* que sea podrá entrar un servidor de reemplazo para suplir a ese servidor caído, en otro caso el servidor o servidores que aún sigan vivos mantendrán la carga entre los restantes.

[22] Significado de Heartbeat en español

[23] Este tiempo es configurado por cada persona dependiendo de sus necesidades.

*Ldirectord*

En el caso del *Ldirectord,* se encarga de monitorear los servidores a través de una petición de *URL* ya conocida verificando que la respuesta de ésta tenga algunos elementos específicos de la aplicación. En el caso de que alguno de los servidores que se encuentren activos llegara a fallar, éste se quitará de la lista de servidores activos y volverá a ser agregado cuando vuelva a estar en funcionamiento. En el caso de que todos los servidores activos se encuentren en fallos, el servidor de fallos será puesto en la lista de los servidores activos hasta que alguno de los servidores que se encontraban en la lista de activos se recupere y se encuentre sin ningún problema y trabajando normalmente.

Después de buscar las opciones para la alta disponibilidad, veremos las herramientas que por *software* nos pueden dar el balanceo de carga, estas herramientas son:

- *Linux Virtual Server LVS*
- *Haproxy*
- *Pound*

*Linux Virtual Server LVS*

Es un servicio que nos permite implementar el balanceo de carga y la alta disponibilidad en uno solo. Este servicio del balanceo de carga lo implementa con una herramienta llamada director, este director será la máquina por la cual los clientes tengan acceso, recibirá las peticiones y las enviará hacia los servidores reales que estén en los archivos de configuración como sus máquinas disponibles para el balanceo, esto lo realizará dependiendo del algoritmo que tenga para distribuir la carga de peticiones.

Se puede tener otra máquina con la cual se realice la alta disponibilidad, ya que en dado caso de que el balanceador falle, las peticiones no se podrán llegar a ver, por lo que este servicio puede incluir la alta disponibilidad y llegar a ser reemplazada por otra máquina la cual se convertirá en el nuevo balanceador de carga.

*Haproxy*

Esta herramienta es muy rápida y bastante confiable que nos ofrece alta disponibilidad, balanceo de carga y un *proxy* para aplicaciones que están basadas en los protocolos *TCP* y *HTTP*, en otras palabras para servidores web. Este servicio es muy recomendable utilizarlo para servidores con sitios web que tengan cargas altas de peticiones, donde es necesario el procesamiento de capa 7. Esta herramienta puede soportar hasta miles de conexiones simultáneas en cientos de instancias, esto sin alterar la estabilidad del sistema. Como esta herramienta es un balanceador por *software*, utiliza el *buffering* TCP, esto es porque el sistema aumenta la cantidad de conexiones pero reduciendo el tiempo de las sesiones, esto lo hace para realizar más conexiones y para poder dejar espacio disponible para más conexiones. Para cada sesión el servicio de *haproxy* necesita a aproximadamente 16 KB, por cada GB de memoria RAM, el servicio soporte 60,000 sesiones, eso claro teóricamente pero en la práctica realmente son 40,000 sesiones.

*Pound*

Este servicio tiene la ventaja de actuar como un *proxy* inverso, un balanceador de carga y un servidor *front-end*[24] para *HTTPS*[25]. Además fue creado para habilitar el

[24] Es la parte del software que interactúa con el o los usuarios.

[25] *Hipertext Transfer Protocol Secure* por sus siglas en inglés, es la combinación del protocolo de navegación por internet *HTTP* y protocolos criptográficos para una sesión de internet más segura.

balanceo de carga entre varios servidores reales y para permitir una conexión *SSL*[26] para servidores web que no lo ofrecen de forma nativa.

*Pound* puede llegar a soportar 30,000 peticiones por día, con un máximo de 600 peticiones por segundo, este servicio ah sido probado en una gran variedad de servidores web, por ejemplo: *Apache, IIS, Zope, WebLogic, Tomcat* e *iPlanet*.

*Pound* tiene una muy buena herramienta la cual puede mantener un seguimiento de las sesiones que se tienen entre los clientes y los servidores reales. Sin embargo como *HTTP* esta definido como un protocolo sin estado, esto complica mucho que esta herramienta funcione adecuadamente, ya que anteriormente algunos otros servicios lo habían tratado de implementar pero ninguno trabajaba de manera óptima.

## 2.5 PROPUESTA DE LA SOLUCIÓN

Una vez que se investigaron las posibilidades con las cuales se pueden trabajar, propuse una infraestructura para la alta disponibilidad, ya que se generaría una pérdida de recursos que sólo una máquina se le agregarán más de un nodo, ya que en la mayoría del tiempo no se encuentra una carga lo suficientemente pesada como para pensar en poner más de un nodo, sin embargo la petición del Profesor Valero es que genere una posibilidad de alta disponibilidad, a esto la propuesta que generé con base a las necesidades y las posibilidades que tenemos en cuanto a recursos, consiste en migrar unos 5 ó 6 servidores con servicios web a varios nodos y generar la alta disponibilidad para éstos, se incrementarían los recursos de estos nodos y gracias a que *Apache* tiene la posibilidad de generar páginas virtuales, podemos agregar varias páginas a un solo nodo, por lo que se crean más nodos con las mismas características para generar la alta disponibilidad.

[26] *Secure Sockets Layer* por sus siglas en inglés, es un protocolo criptográfico que proporcionan comunicaciones seguras por una red.

### 2.5.1 RECURSOS MÍNIMOS

Teniendo en cuenta que se va a crear la alta disponibilidad a más de un servicio web, se necesita una gran cantidad de procesamiento para que el balanceador acepte todas las peticiones y pueda mandarlas a los nodos que contienen la información, estos nodos tienen que soportar todas las peticiones que le lleguen y poder atenderlas según el método de balanceo que se vaya a configurar. Para que los nodos tengan las condiciones siguientes se tendrán que contar con mínimo 8 GB[27] de memoria RAM y con dos núcleos de procesamiento, las máquinas deberán de tener el sistema operativo *CentOS*[28] (versión 6.5) para que tengan una buena compatibilidad entre ellas y la instalación de los servicios sea la misma y no genere retraso en la instalación.

### 2.5.2 PROPUESTA DE UN MODELO PARA LA SOLUCIÓN[29]

En la figura 2.2 se muestra el modelo que se utilizará para la generación de la alta disponibilidad y el balanceo de carga, se necesitarán 5 máquinas virtuales como se muestra en el modelo, de estas sólo 4 serán nodos y todos estarán activos debido a que en un futuro no sólo se utilizará para un *moodle*, este modelo podrá albergar hasta siete u ocho *mooodles* que sean pequeños o medianos. La quinta máquina virtual se encargará del balanceo de carga y de la alta disponibilidad, la cual va a facilitar que los servidores activos no se saturen con las peticiones y tengan que hacer el servicio de balanceo y ver si los otros nodos se encuentran disponibles, si no que esta quinta máquina se dedicará únicamente al balanceo y a verificar si los nodos

[27] Abreviación de gigabyte

[28] Sistema operativo linux basado en Red Hat

[29] Este modelo servirá para la migración de varios servidores, pero por el momento solo se mostrará el caso del servidor de apuntes en línea.

están activos o no, liberando recursos. Se necesitarán un total de 5 direcciones ip que estén dentro de la misma red, las cuales se dividirán de la siguiente manera:

- 1 dirección ip para el servidor de balanceo y alta disponibilidad.
- 4 direcciones para los nodos.

De esta manera se podrá garantizar que los servidores estarán siempre activos para cuando se les necesite, ya que los otros factores que podrían llegar a ser un problema serían:

- Problemas eléctricos.
- Problemas en la red.

Pero gracias a la planta de emergencia con la que cuenta el departamento y los nuevos ajustes en la red estos problemas pasan a ser descartados, preocupados solamente por el caso del estado y procesamiento de los servidores.

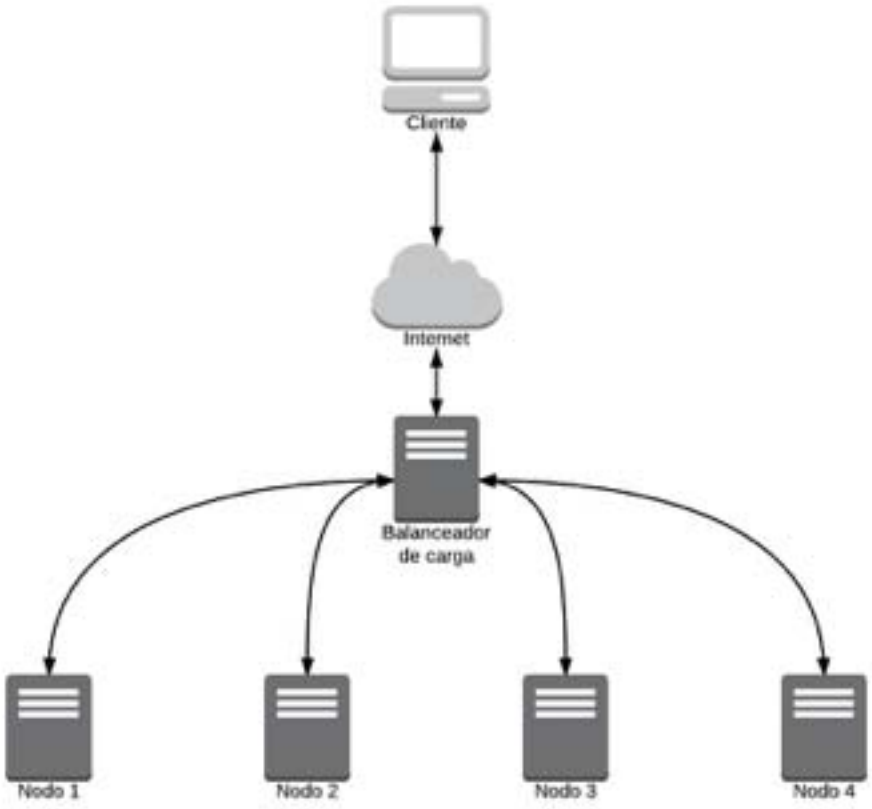



Figura 2.2 Modelo de la solución al problema de alta disponibilidad.

Resumen del capítulo:

Al final de este capítulo podemos entender la problemática que se tiene en la Facultad de Veterinaria Medicina y Zootecnia en cuanto a la disponibilidad de no solamente uno de sus servidores si no en todos, la necesidad de una mejor infraestructura y organización de ésta para que los servicios que brinda la facultad estén siempre disponibles.

El profesor Valero empezó con esta necesidad de brindar una estabilidad a su servidor pero viéndolo desde su punto de vista, no únicamente debería ser el suyo si no que también deberían de ser todos los demás servidores, por eso la necesidad de implementar un modelo que si bien no va a poder albergarlos a todos, se puede usar como pionero para después migrar al resto, a este siguiente paso de la alta disponibilidad.

Encontramos también en este capítulo las opciones que se consideraron dentro de los posibles candidatos para llegar a esta solución, balanceadores y generadores de alta disponibilidad que a lo largo de la búsqueda de información fueron los que se encontraron con mejores resultados en cuanto a opiniones y soporte de parte de sus desarrolladores.

# CAPÍTULO III

# IMPLEMENTACIÓN DE LA SOLUCIÓN

Entendiendo las limitaciones con las que cuenta el Departamento de Cómputo de la Facultad de Medicina Veterinaria y Zootecnia, la solución deberá de generar la menor cantidad de procesamiento, para garantizar que puedan recibir la mayor cantidad de peticiones sin que los nodos tengan problemas para responderlas con la mayor prontitud, para esto el servidor que generará la alta disponibilidad y el balanceo de carga deberá de tener los servicios indispensables para que pueda dirigir las peticiones y verificar la disponibilidad de los nodos para cumplir con estas peticiones.

## 3.1 *HAPROXY*

Como se comentó en el capítulo anterior este servicio puede proporcionar el balanceo de carga y al mismo tiempo la alta disponibilidad, debido a que se necesita la mayor cantidad de recursos, se decidió que ésta sería la mejor opción dado que no se gastarían tantos recursos y la compatibilidad entre las aplicaciones no sería un tema de preocupación a la hora de la instalación. Otro de los puntos a favor es que en la búsqueda para elegir la mejor aplicación, se encontró que esta herramienta se puede descargar desde los repositorios que tiene el sistema operativo *CentOS*, esto también nos benefician en cuanto a la compatibilidad entre el sistema operativo y la aplicación; así mismo se tiene un soporte en dado caso que se tenga algún problema con el servicio a instalar y se cuenta con mucha información de la instalación y configuración de esta aplicación.

### 3.1.1 INSTALACIÓN DEL SERVICIO DE BALANCEO DE CARGA Y ALTA DISPONIBILIDAD

Para la instalación del servicio se necesita tener permisos de administrador o con los privilegios necesarios para la instalación, en la figura 3.1 se ve un ejemplo donde se puede notar que se tienen los permisos de administrador.


Figura 3.1 Terminal de administrador

Como se puede ver viene la estructura *roto*@prueba1, el *root* significa qué tipo de usuario se usa, en este caso administrador y prueba1 es parte del nombre del servidor.

Una vez que se ha identificado que se tienen los permisos correspondientes se busca el paquete *haproxy*, esto se hace con el comando, que tiene el sistema operativo, llamado *yum*, en la figura 3.2 se muestra la forma de buscar.

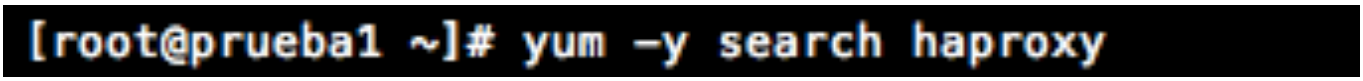



Figura 3.2 Búsqueda por el comando *yum*

Con el comando *yum* se pueden buscar actualizaciones del sistema, programas que se encuentran dentro de los repositorios del sistema operativo y remover los que ya no se quieren. Ahora, en la figura 3.3 se observa el resultado de la búsqueda anterior:

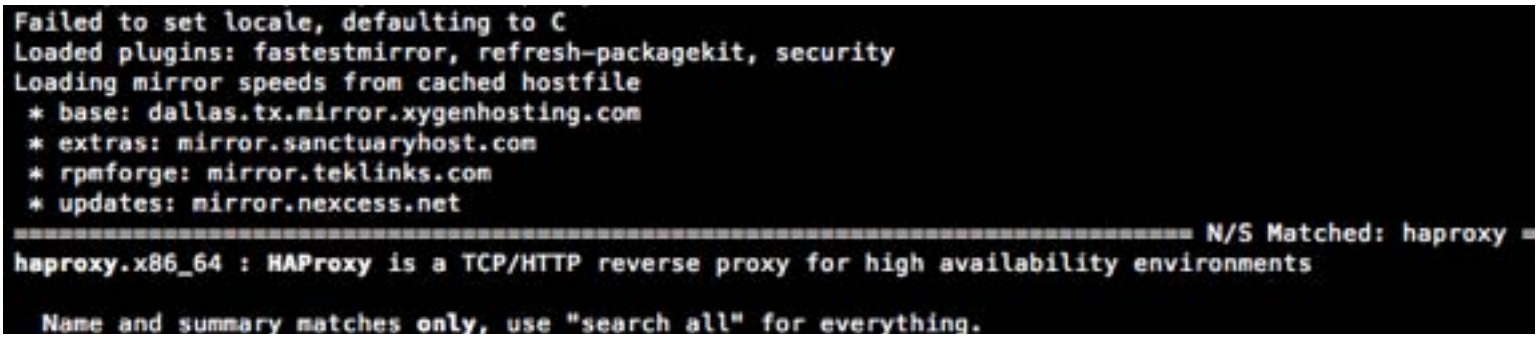



Figura 3.3 Resultados de la búsqueda del programa

Lo que se muestra en la figura 3.3 empieza cargando algunos repositorios del sistema operativo, después viene donde el comando va a buscar los programas y las actualizaciones y después de la doble línea se tienen los resultados de la búsqueda, el comando no solamente busca el programa si no todo lo que venga relacionado con él, en este caso no hay ningún otro paquete relacionado, pero en algunos casos viene

paquetería adicional o programas que por algún motivo estén relacionados con el motivo de búsqueda.

Para la instalación de los paquetes se volverá a utilizar el comando yum, en la figura 3.4 se mostrará el comando.

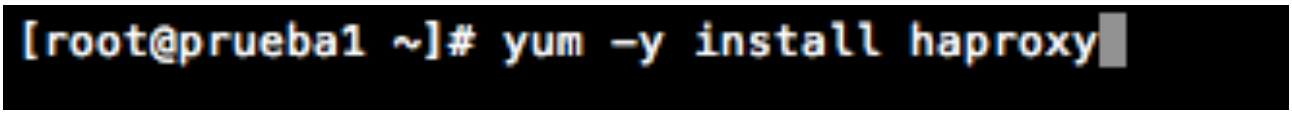



Figura 3.4 Comando de instalación del programa *haproxy*

Como se observa en la figura anterior al comando yum se le agregó la opción install, ésta es la que nos indica que vamos a instalar el servicio *haproxy* y la opción -y se utiliza para decir sí a todas las posibles preguntas que hace el comando yum, por ejemplo en algunos momentos pregunta si se está seguro de la instalación del programa; para evitar esos problemas se utiliza la opción -y.

En la figura 3.5 se puede ver cómo la instalación del paquete de *haproxy* al igual que con la búsqueda empieza por los repositorios del sistema; verifica que el paquete está disponible para la descarga. El comando descargará el paquete con la arquitectura necesaria para el sistema y por último lo instalará, al final de la figura 3.5 se observa que efectivamente lo instaló de manera correcta, si el paquete necesitara alguna otra paquetería adicional para su funcionamiento al final hubiera mostrado todos los paquetes instalados.

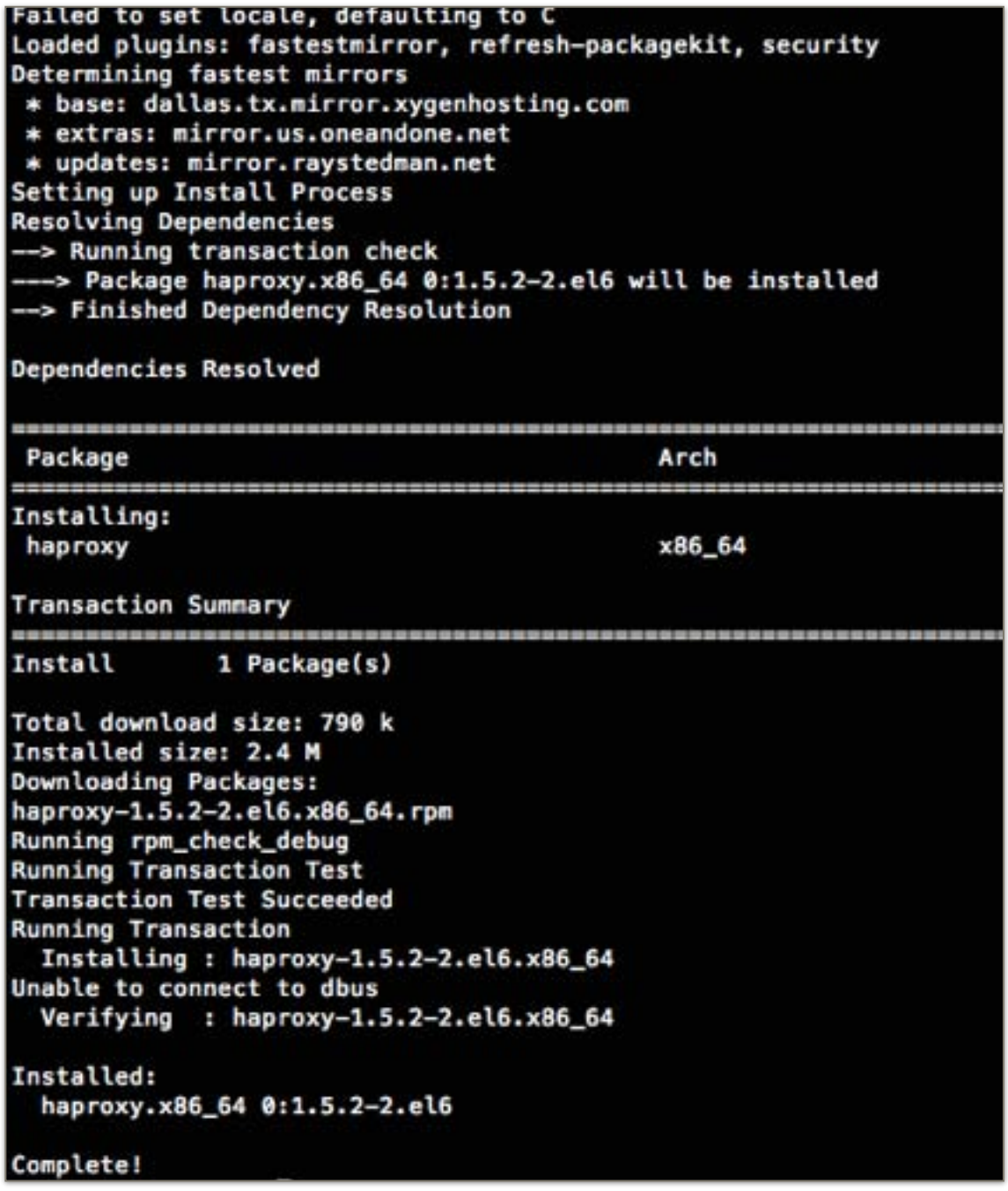



Figura 3.5 Instalación del servicio *haproxy*

Para comprobar que la instalación se realizó de la manera correcta se busca una carpeta y un archivo, los cuales indicarán que efectivamente la instalación no tuvo problemas.

La carpeta se llamará igual que el servicio *haproxy* dentro de ella habrá un archivo llamado haproxy.cfg. Por último el archivo se llama igual *haproxy;* este se encuentra dentro de la carpeta /etc/init.d, ambos se pueden listar con el comando ls, en la imagen 3.6 se da un ejemplo de la forma de utilizar el comando ls para listar el contenido de una carpeta.


Figuro 3.6 Ejemplo de listado de carpetas

### 3.1.2 CONFIGURACIÓN DEL SERVICIO *HAPROXY*

Una vez que se comprobó que la instalación se realizó de manera correcta, lo siguiente será la configuración del archivo llamado haproxy.cfg, en la imagen 3.7 se usa el comando vi, este comando sirve para la edición de archivos[30], con el cual se verá el archivo que se necesita al modificarse para que el balanceo y la alta disponibilidad se realice de manera adecuada.


Figura 3.7 Comando vi utilizado para la edición del archivo haproxy.cfg

[30] Se necesitan permisos de lectura y escritura, no es necesario los de administrador.

Lo que mostrará el resultado será lo que se ve en las figuras 3.8, 3.9 y 3.10; en la figura 3.8 lo que se observará es el inicio del archivo donde vendrá el lugar de instalación de las librerías del servicio, donde se almacena la información del proceso cuando el servicio está activo, el número máximo de conexiones, el usuario y grupo correspondientes al servicio.

En la figura 3.9 vienen los valores por defecto del balanceo, muestra lo que se va a balancear; en el caso http, el número de intentos antes de descartar una petición, los siguientes son los tiempos que va a utilizar para las peticiones y las comunicaciones con los nodos, así como el número máximo de conexiones que nuestro servidor va a poder recibir al mismo tiempo.

En la figura 3.10 viene una muestra de cómo se tiene que realizar la configuración del balanceo, tanto el método de balanceo, como la forma de agregar los servidores, para que el servicio los reconozca como sus nodos.

```
global
    # to have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
    # 1) configure syslog to accept network log events.  This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
    #
    # 2) configure local2 events to go to the /var/log/haproxy.log
    #   file. A line like the following can be added to
    #   /etc/sysconfig/syslog
    #
    #    local2.*                       /var/log/haproxy.log
    #
    log         127.0.0.1 local2

    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon
```

Figura 3.8 Inicio del archivo haproxy.cfg


Figura 3.9 Segunda parte del archivo harpoxy.cfg

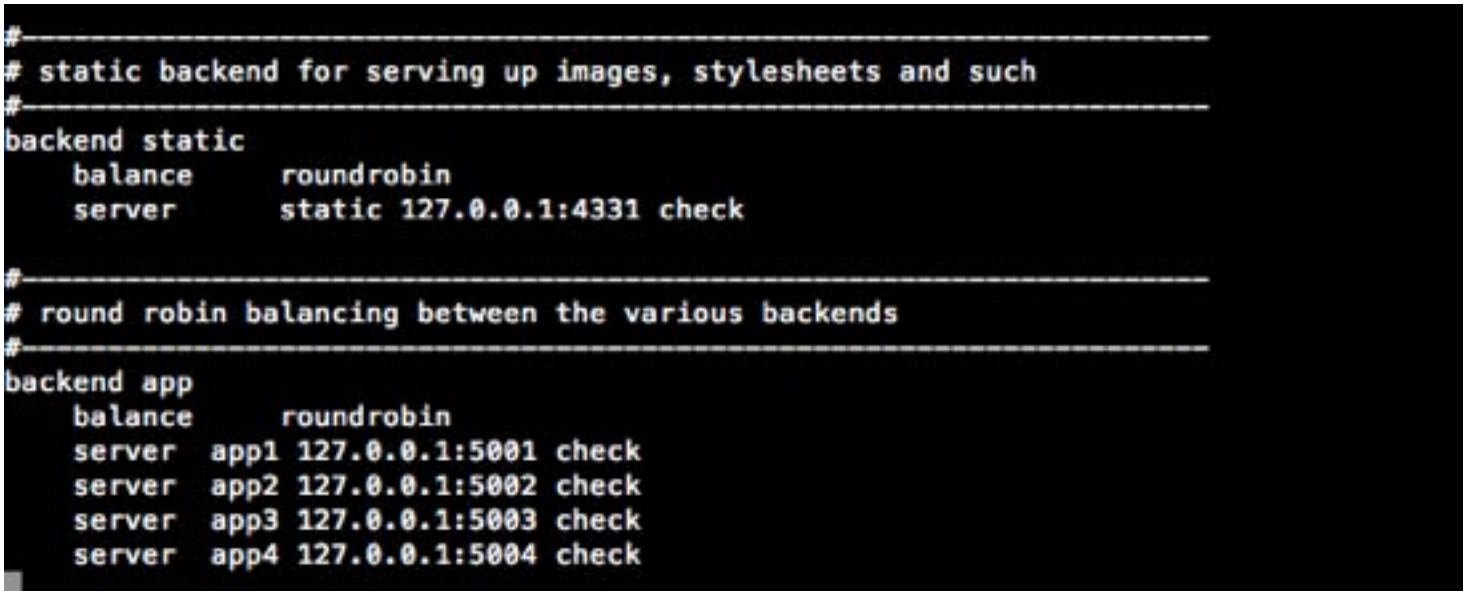



Figura 3.10 Tercera parte del archivo haproxy.cfg

Una vez ubicado lo necesario para la configuración, sólo se modificará la parte de los tiempos, los servidores y se le agregarán unas líneas adicionales que durante la investigación, para la correcta instalación del servicio, no tuvo ningún error. Para empezar a modificar el archivo, presionaremos la tecla i y donde se muestre un rectángulo del color del texto es dónde se encuentra el cursor para la modificación de este archivo.

La primera parte del documento no se debe de modificar a excepción de la máxima cantidad de conexiones que también se volverá a ajustar en la segunda parte del archivo. En la figura 3.11 se puede notar que no hubo cambio. En la segunda parte se va a configurar los tiempos que el servicio *haproxy* tomará como referencia para el término de una conexión o de que algún nodo pueda estar inactivo. A continuación se describirá cada uno de los segmentos ajustables según las necesidades de cada quien:

- *retries*[31], son los intentos que el servidor realizará para la conexión con los nodos una vez que haya alguna solicitud.
- *timeout http-request*[32], es el tiempo límite antes de que el servidor descarte la petición web.
- *timeout queue*[33], es el tiempo en que la petición estará en la cola de espera antes de ser descartada.
- *timeout connect*[34], es el tiempo máximo que tendrá que esperar el servidor para hacer la conexión con alguno de los nodos una vez que este haya recibido alguna petición.

[31] Número de intentos

[32] Tiempo límite de la solicitud http.

[33]Tiempo límite en la cola.

[34]Tiempo límite de conexión.

- *timeout client*[35], se refiere al tiempo límite que tendrá que esperar el cliente antes de que descarte la devolución de la solicitud web.
- *timeout server*[36], es el tiempo máximo que va a esperar conectar con alguno de los nodos antes de que el servidor lo elimine de su lista de opción al balanceo.
- *timeout http-keep-alive*[37], es el tiempo máximo que el servidor preguntará si el servicio de *http* se encuentra disponible.
- *timeout check*[38], este tiempo es cada cuanto el servidor verificará si los nodos se encuentran activos y están disponibles para recibir peticiones.
- *maxconn*, es el número máximo de conexiones que el servidor podrá llegar a recibir.

[35] Tiempo límite de conexión al cliente

[36] Tiempo límite del servidor.

[37] Tiempo límite de la actividad http.

[38] Tiempo de chequeo.

```
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           30s
    timeout connect         10s
    timeout client          1m
    timeout server          30s
    timeout http-keep-alive 10s
    timeout check           10s
    maxconn                 6000
```

Figura 3.11 Cambios en la segunda parte del archivo[39].

```
#balanceo

listen prueba.com 132.248.62.33:80

mode http
stats enable
balance roundrobin
cookie JSESSIONID prefix
option httpclose
option forwardfor

option httpchk HEAD / check.txt HTTP/1.0

server prueba1.fmvz.unam.mx 132.248.62.36:80 cookie A check
server localhost.localdomain 132.248.62.45:80 cookie B check
server localhost.localdomain 132.248.62.37:80 cookie C check
server localhost.localdomain 132.248.62.43:80 cookie D check
```

Figura 3.12 Configuración del balanceo

[39] Estos tiempos fueron usados para este caso, se debe modificar conforme a la necesidad de cada institución.

Para la parte del balanceo en la figura 3.12 se mostrará el ejemplo de los cambios necesarios para el balanceo, se va a empezar la configuración con *listen* donde su traducción al español es escuchar, por lo tanto el comando dirá que estará al pendiente de las solicitudes. Lo siguiente, será poner el nombre de la página o servidor; se dejará un espacio y por último la dirección ip del balanceador de carga, agregando dos puntos especificando si por algún puerto, determinado, el servidor recibirá las solicitudes. Para continuar se realiza un salto de línea y se agrega *mode http,* esto es lo que vamos a balancear, para este caso es *http*, lo siguiente a poner es *stats* [40], el cual va a referir al estado del balanceador, para activarlo se deja en el estado *enable*[41]. El siguiente campo es muy importante ya que es el tipo de algoritmo que se va a utilizar, haproxy cuenta con 4 tipos de algoritmos los cuales son:

- *RoundRobin.*
- Por pesos.
- Menos conexiones.
- Fuente.

Dependiendo el tipo de nodos que se tengan será el tipo de algoritmo a utilizar, ya que son diferentes en cuanto a cómo se van a distribuir las solicitudes, en el caso de *roundrobin* se distribuyen de manera equitativa, pero para realizarlo, los nodos deben de tener las mismas características.

El balanceo por pesos se refiere a las características del sistema, puede que alguno de los nodos tenga mejores características que otro, por lo tanto el nodo con mejores características recibirá más solicitudes que los demás y los otros recibirán solicitudes dependiendo de sus capacidades.

[40] Diminutivo de status que quiere decir estado.

[41] Palabra en inglés para decir activado.

El balanceo por menos conexiones, como su nombre lo dice buscará dentro de los nodos al nodo que tenga menos solicitudes y las mandará hasta que los otros nodos terminen y se vuelvan los nodos con menos conexiones.

Por fuente, esté es de los menos eficientes ya que consiste en utilizar la dirección ip para generar un *hash*[42] y va a tener conexión con alguno de los nodos no importando cuál de ellos responda, si esa misma dirección ip vuelve a realizar otra solicitud el mismo nodo que le contestó por primera vez tiene la obligación de responderle; este método se vuelve eficiente porque se puede cargar a un nodo más que otro bajando el rendimiento del balanceo.

Por parte de la propuesta era que todos los nodos contaran con las mismas características, por lo tanto la mejor opción es el algoritmo del balanceo por *roundrobin*.

Lo siguiente para la configuración es un identificador, para esto, se realizará con la línea *cookie*[43] *JSESSIONID prefix*[44], este identificador se necesita para la comunicación entre el nodo y el cliente, esto lo hace marcando cada *cookie* de sesión con el identificador que se le va a agregar y un identificador de cada nodo[45], al final *prefix* con lo que el identificador se compondrá con *JSESSIONID* y el prefijo de cada nodo.

Lo siguiente es para que el cliente y el servidor cierren directamente después de la primera solicitud, la opción *fowardfor* es una función que *haproxy* hace para añadir a cada solicitud un encabezado de respuesta, eso para que el nodo le mande una respuesta a la petición del cliente.

---

[42] Son un tipo de funciones que a la entrada de algún dato, contraseña o palabra, devuelven una salida de caracteres alfanumérico de longitud fija que representa lo mismo que el dato,contraseña o palabra.

[43] Información mandada por un sitio web al navegador del usuario, esto para que el sitio web pueda consultar actividad previa.

[44] Es la traducción de prefijo al inglés.

[45] Este identificador se podrá ver más adelante cuando sean definidos los nodos en la página siguiente.

La última parte de estas opciones es para la alta disponibilidad ya que se va a agregar el *httpchk*, esta opción lo que va a hacer es que se le va a poner una ruta donde buscará un archivo en especifico, este archivo no importa si no tiene nada solo tiene que tener el nombre que nosotros hayamos definido para que en este caso el archivo se llamará check.txt[46] y estará localizado en el directorio / de nuestro servidor.

Para finalizar con las configuraciones se van a definir los nodos que van a ser utilizados dentro del balanceo de carga, se va a iniciar con la palabra *server*[47] a continuación se dejará un espacio, después se pondrá el nombre de nuestro servidor, seguido de otro espacio y se continuará con la dirección ip de cada servidor y el puerto por el cual cada uno tendrá que escuchar y al final se dejará el identificador que habíamos visto que se necesitaba en la opción de la *cookie* en este caso serán A,B,C y D.

Se guardará el archivo, para esto presionaremos la tecla ESC para salir del modo de edición y para guardar pondremos los siguiente :wq!, saldremos del archivo pero los cambios en él se habrán realizado de manera correcta.

Solo nos haría falta iniciar el servicio, lo haremos con el siguiente comando service *haproxy start*, con esto nuestro servicio dirá *done* esto significará que está activado y listo para utilizarse.

### 3.1.3 PRUEBA DE BALANCEO[48]

Las pruebas de balanceo serán muy fáciles de realizar, una vez que la configuración esté lista y el servicio activado, vamos a ir hasta nuestro navegador y colocaremos la

---

[46] Este archivo podrá llamarse de cualquier otra manera con la única condición que sea igual para todos los nodos y que sea escrito de la misma manera en el archivo de configuración del *haproxy*.

[47] Traducción de servidor en inglés.

[48] Para la prueba los nodos y el balanceador deberán tener activo el servicio de http, si no aún no está instalado en el apartado 3.2.1 se mostrará cómo instalar el servicio apache desde línea de comandos.

dirección ip de nuestro balanceador de carga, para poder ver los cambios, se van a editar los archivos índex.html y a cada nodo se le va a quitar el título original por cada uno de los servidores, así que cada nodo tendrá el título server más el número 1, 2, 3 y 4, el número de cada server puede ser indiferente a la dirección ip, solo se van a nombrar para poder identificarlos a la hora del balanceo, en las figuras 3.13, 3.14, 3.15 y 3.16 se va  ha mostrar el cambio del índex.html para que sirva como ejemplo.

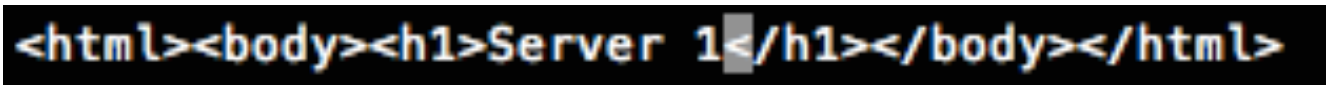



Figura 3.13 Primer servidor modificado en su index.html.

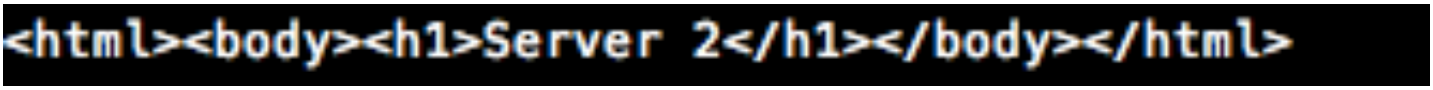



Figura 3.14 Segunda servidor modificado en su index.html


Figura 3.15 Tercer servidor modificado en su index.html


Figura 3.16 Cuarto servidor modificado en su index.html

Estos cambios van a servir para identificar a la hora de comprobar la configuración de nuestro servidor de balanceo de carga, como se mencionó en el capítulo 1, el balanceo de carga no va a poder ser detectado por los usuarios finales, por eso con este fin se hicieron las anteriores modificaciones, en la figura 3.17 se va a mostrar el resultado de poner la dirección ip de nuestro balanceador de carga en algún navegador.


Figura 3.17 Ejemplo de los resultados del balanceo de carga.

Como se puede notar aparece el nombre de server 1, que fue una de las modificaciones, pero si volvemos a recargar la página unas cuantas veces[49] sucede otro cambio como se muestra en la figura 3.18.

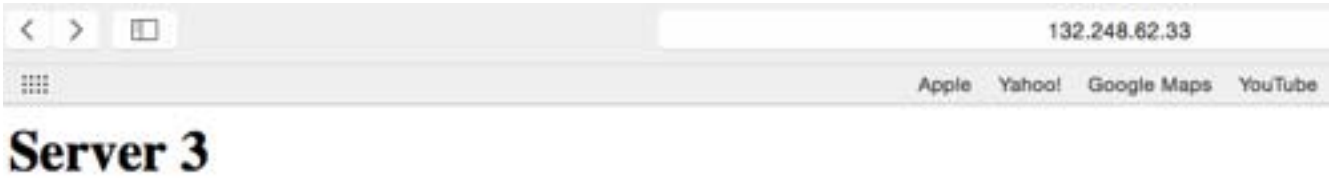



Figura 3.18 Recarga de la página del balanceador de carga.

[49] Esto se hace por el tipo de balanceo de carga, como es *roundrobin* tenemos que hacer que nuestra petición llegue a otro nodo diferente al que ya nos respondió anteriormente.

En esta ocasión salió el server 4, puede ser que así sea la modificación del segundo nodo, si volvemos a recargar la página una cuantas veces más como se mostrará en la figura 3.19 y 3.20 se observara que salen los demás nodos, solo que no saldrán en orden y se podrán repetir, esto debido a que como los nodos responden la petición vuelven a estar disponibles para recibir más peticiones, por este motivo el usuario final no llegaría ver la diferencia cuando la página como lo es apuntes en línea llegará a tener un balanceador atrás de ella.


Figura 3.19 Recarga de la página del balanceador de carga por segunda vez.

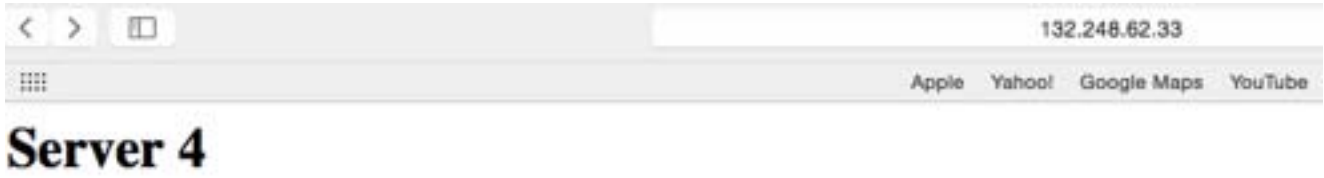



Figura 3.20 Recarga de la página del balanceador de carga por tercera vez.

## 3.2 MIGRACIÓN DE LOS SERVICIOS WEB NECESARIOS

Como se comentó en el capítulo 2, el propósito claro de la implementación de un servidor de carga con alta disponibilidad, era la migración de la plataforma web llamado *moodle* el cual nos sirve para poder tomar cursos a distancia, entrega de tareas, exámenes, actividades, etc. Pero esta plataforma requiere elementos específicos; *moodle* es una plataforma web, pero no está hecho para que fuera una página que se pudiera desarrollar por sí misma, necesita de un servicio web, en este caso será el servicio de apache el cual ayudará como servicio web adicional, aunado a esto, la plataforma está basada en el lenguaje de programación *php*, por esto es indispensable que todos nuestros nodos lo tengan. Otra parte muy importante de la plataforma *moodle* es la base de datos, ya que se pueden realizar exámenes, tareas y otras actividades cada alumno tendrá una cuenta asociada la cual se guardará también en nuestros nodos, este servicio también tendrá que ser instalado, en este caso se utilizará el servicio de *mysql* para las bases de datos.

Es importante tener en cuenta que los servicios que anteriormente se acaban de describir, a excepción de *php,* pueden variar, debido a que existen diferentes tipos de servicios web que pueden utilizar *moodle,* así como de bases de datos, no son necesarios los que se presentan en este trabajo. Si es el primera vez que instalan la plataforma *moodle* se pueden utilizar, pero si es una migración como en este caso, es necesario utilizar los servicios que tenemos en nuestro *moodle* que se desea migrar.

### 3.2.1 *APACHE*

Este es un servicio web basado en *http*, y es una de las herramientas que se tienen que ocupar para la migración de la plataforma *moodle*. Para la instalación y la configuración de nuestro servicio de *apache*, se necesitan tener privilegios de administrador o los privilegios necesarios.

Para iniciar se va crear una carpeta, la cual podrá ser llamada de cualquier forma, en este caso se llamará sw, se entrara a la carpeta para ahí empezar a trabajar con la instalación del servicio apache. En la figura 3.21 se muestran los comando para crear la carpeta y para entrar en ella.

```
[root@localhost ~]# mkdir /sw
[root@localhost ~]# cd /sw
```

Figura 3.21 Comando de creación de carpeta y entrar en ella.

Se va a necesitar el comando wget, el cual servirá para extraer de una página algún archivo que nosotros vayamos a utilizar, así que hay que asegurarse de tenerlo[50].

En las figuras 3.22, 3.23, 3.24, 3.25, 3.26, 3.27, 3.28 y 3.29 se mostrarán una serie de paquetes que serán descargados, los cuales son necesarios para la instalación del servicio web apache.

```
[root@localhost sw]# wget http://apache.mirrors.hoobly.com//apr/apr-1.5.1.tar.gz
```

Figura 3.22 Comando wget utilizado para descargar el paquete apr.

[50] En caso de no contar con el comando wget, se tendrá que instalar utilizando el comando yum de la siguiente manera yum -y install wget.

```
--2014-12-03 08:54:56--  http://apache.mirrors.hoobly.com//apr/apr-1.5.1.tar.gz
Resolving apache.mirrors.hoobly.com... 66.160.172.98
Connecting to apache.mirrors.hoobly.com|66.160.172.98|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1020833 (997K) [application/x-gzip]
Saving to: “apr-1.5.1.tar.gz”

100%[======================================================================>] 1,020,833    358K/s   in 2.8s

2014-12-03 08:55:02 (358 KB/s) - “apr-1.5.1.tar.gz” saved [1020833/1020833]
```

Figura 3.23 Descarga del paquete apr.

```
[root@localhost sw]# wget http://apache.mirrors.hoobly.com//apr/apr-util-1.5.4.tar.gz
```

Figura 3.24 Comando wget para descargar el paquete apr-util.

```
--2014-12-03 08:55:58--  http://apache.mirrors.hoobly.com//apr/apr-util-1.5.4.tar.gz
Resolving apache.mirrors.hoobly.com... 66.160.172.98
Connecting to apache.mirrors.hoobly.com|66.160.172.98|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 874044 (854K) [application/x-gzip]
Saving to: “apr-util-1.5.4.tar.gz”

100%[======================================================================>] 874,044      328K/s   in 2.6s

2014-12-03 08:56:01 (328 KB/s) - “apr-util-1.5.4.tar.gz” saved [874044/874044]
```

Figura 3.25 Descarga del paquete apr-util.

```
[root@localhost sw]# wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.36.tar.gz
```

Figura 3.26 Comando wget para descargar el paquete pcre.

```
--2014-12-03 08:57:06--  ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-8.36.tar.gz
           => “pcre-8.36.tar.gz”
Resolving ftp.csx.cam.ac.uk... 131.111.8.115
Connecting to ftp.csx.cam.ac.uk|131.111.8.115|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD (1) /pub/software/programming/pcre ... done.
==> SIZE pcre-8.36.tar.gz ... 2009464
==> PASV ... done.    ==> RETR pcre-8.36.tar.gz ... done.
Length: 2009464 (1.9M) (unauthoritative)

100%[=====================================================================>] 2,009,464    196K/s   in 13s

2014-12-03 08:57:31 (147 KB/s) - “pcre-8.36.tar.gz” saved [2009464]
```

Figura 3.27 Descarga del paquete pcre.

```
[root@localhost sw]# wget http://mirror.tcpdiag.net/apache//httpd/httpd-2.4.10.tar.gz
```

Figura 3.28 Comando wget para descargar el paquete httpd.

```
--2014-12-03 08:58:29--  http://mirror.tcpdiag.net/apache//httpd/httpd-2.4.10.tar.gz
Resolving mirror.tcpdiag.net... 69.160.243.150
Connecting to mirror.tcpdiag.net|69.160.243.150|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6820719 (6.5M) [application/x-gzip]
Saving to: “httpd-2.4.10.tar.gz”

100%[=====================================================================>] 6,820,719   1.09M/s   in 6.2s

2014-12-03 08:58:36 (1.06 MB/s) - “httpd-2.4.10.tar.gz” saved [6820719/6820719]
```

Figura 3.29 Descarga del paquete httpd.

Estos paquetes son necesarios para la instalación del servicio apache, para comprobar que están en la carpeta sw, como ya estamos adentro se ejecutará el comando ls, en la figura 3.30 se verá el resultado del comando.

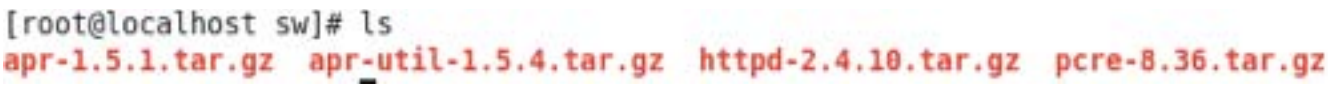



Figura 3.30 Resultado del comando ls.

Lo siguiente a realizar será la instalación, para ello se deben de instalar los paquetes de una manera en específico, no en cualquier orden, se deben instalar apr, apr-util, pcre y httpd, en este orden para que no genere ningún error y la instalación se complete.

La siguiente forma de instalación se realizará para generar un poco de más seguridad en el servidor y para saber dónde se pueden encontrar los paquetes que se instalarán.

Se comenzará con apr, para esto se descomprime el paquete con el comando tar, en la figura 3.31 se mostrará la forma de ejecutar el comando, una vez que se haya descomprimido el paquete hay que entrar  a la carpeta con el comando cd, espacio y el nombre apr, una vez dentro, teclea el comando ls para listar el contenido, en la figura 3.32 se mostrará el contenido. Después de eso entrar al modo de edición de archivos, en este caso en un archivo llamado configure, en la figura 3.33 se puede ver el comando vi junto con el archivo a editar, buscar en el archivo la línea que diga $RM “$cfgfile”, y se modificará como se muestra en la línea 3.34 esto debido a que si no se realiza la paquetería de apr no podrá ser instalada; una vez que se haya modificado la línea correspondiente hay que salir del modo de edición presionando ESC y se guardara con el siguiente comando :wq!, salir a la carpeta del paquete apr y

comenzar con la instalación, en la figura 3.35, 3.36, 3.37, 3.38, 3.39 y 3.40 se mostrará el proceso de instalación, así como las salidas de los comandos para que se verifique que la instalación fue correcta.

```
[root@localhost sw]# tar -zxvf apr-1.5.1.tar.gz
```

Figura 3.31 Descompresión del paquete apr.

```
[root@localhost apr-1.5.1]# ls
apr-config.in  build             configure.in  libapr.dep    memory       random       threadproc
apr.dep        buildconf         docs          libapr.dsp    misc         README       time
apr.dsp        build.conf        dso           libapr.mak    mmap         README.cmake tools
apr.dsw        build-outputs.mk  emacs-mode    libapr.rc     network_io   shmem        user
apr.mak        CHANGES           encoding      LICENSE       NOTICE       strings
apr.pc.in      CMakeLists.txt    file_io       locks         NWGNUmakefile support
apr.spec       config.layout     helpers       Makefile.in   passwd       tables
atomic         configure         include       Makefile.win  poll         test
```

Figura 3.32 Salida del comando ls.

```
[root@localhost apr-1.5.1]# vi configure
```

Figura 3.33 Edición del archivo configure.

```
cfgfile="${ofile}T"
trap "$RM \"$cfgfile\"; exit 1" 1 2 15
$RM -f "$cfgfile"
```

Figura 3.34 Edición de la línea RM.

```
[root@localhost apr-1.5.1]# ./configure
```

Figura 3.35 Ejecución del archivo modificado.

```
configure: creating ./config.status
config.status: creating Makefile
config.status: creating include/apr.h
config.status: creating build/apr_rules.mk
config.status: creating build/pkg/pkginfo
config.status: creating apr-1-config
config.status: creating apr.pc
config.status: creating test/Makefile
config.status: creating test/internal/Makefile
config.status: creating include/arch/unix/apr_private.h
config.status: executing libtool commands
config.status: executing default commands
```

Figura 3.36 Resultados de la ejecución.

```
[root@localhost apr-1.5.1]# make
```

Figura 3.37 Compilación del paquete apr.

```
gcc -E -DHAVE_CONFIG_H  -DLINUX -D_REENTRANT -D_GNU_SOURCE   -I./include -I/sw/apr-1.5.1/include/arch/
unix -I./include/arch/unix -I/sw/apr-1.5.1/include/arch/unix -I/sw/apr-1.5.1/include -I/sw/apr-1.5.1/i
nclude/private -I/sw/apr-1.5.1/include/private  export_vars.c | sed -e 's/^\#[^!]*//' | sed -e '/^$/d'
 >> apr.exp
sed 's,^\(location=\).*$,\1installed,' < apr-1-config > apr-config.out
sed -e 's,^\(apr_build.*=\).*$,\1/usr/local/apr/build-1,' -e 's,^\(top_build.*=\).*$,\1/usr/local/apr/
build-1,' < build/apr_rules.mk > build/apr_rules.out
make[1]: Leaving directory `/sw/apr-1.5.1'
```

Figura 3.38 Salida de la compilación del paquete.

```
[root@localhost apr-1.5.1]# make install
```

Figura 3.39 Instalación del paquete apr.

```
/usr/bin/install -c -m 644 apr.exp /usr/local/apr/lib/apr.exp
/usr/bin/install -c -m 644 apr.pc /usr/local/apr/lib/pkgconfig/apr-1.pc
for f in libtool shlibtool; do \
            if test -f ${f}; then /usr/bin/install -c -m 755 ${f} /usr/local/apr/build-1; fi; \
        done
/usr/bin/install -c -m 755 /sw/apr-1.5.1/build/mkdir.sh /usr/local/apr/build-1
for f in make_exports.awk make_var_export.awk; do \
            /usr/bin/install -c -m 644 /sw/apr-1.5.1/build/${f} /usr/local/apr/build-1; \
        done
/usr/bin/install -c -m 644 build/apr_rules.out /usr/local/apr/build-1/apr_rules.mk
/usr/bin/install -c -m 755 apr-config.out /usr/local/apr/bin/apr-1-config
```

Figura 3.40 Salida de la instalación del paquete apr

Una vez instalado del paquete salir de la carpeta de apr, para instalar los paquetes restantes, con el comando cd .. y se realizarán pasos similares a los anteriores para la instalación; serán casi los mismos a excepción de editar el archivo **configure**, editar una línea de él y en caso de apr-util se modificará la ejecución del **configure**.

Seguiremos con el apr-util, en las figuras 3.41, 3.42, 3.43, 3.44, 3.45, 3.46 y 3.47 se mostrarán los pasos a seguir al igual que con apr, apr-util se va a configurar, compilar y por último a instalar, pero como este paquete requiere de manera forzosa que apr esté instalado, en la parte de la configuración se debe de agregar dónde se encuentra el paquete apr que recién se ha instalado. Para esto cada paquete que se instala se coloca en ciertos lugares en específico, como fue instalado de forma personalizada se va a colocar en la ruta /usr/local y estará dentro de la carpeta apr, sólo se debe de agregar a la configuración la ruta de la carpeta acompañado de la opción —with-apr=, con esto se podrán seguir los pasos similares como a continuación se van a mostrar:

```
[root@localhost sw]# tar -zxvf apr-util-1.5.4.tar.gz
```

Figura 3.41 Descompresión del paquete apr-util.

```
[root@localhost apr-util-1.5.4]# ./configure --with-apr=/usr/local/apr/
```

Figura 3.42 Configuración del paquete apr-util.

```
configure: creating ./config.status
config.status: creating Makefile
config.status: creating export_vars.sh
config.status: creating build/pkg/pkginfo
config.status: creating apr-util.pc
config.status: creating apu-1-config
config.status: creating include/private/apu_select_dbm.h
config.status: creating include/apr_ldap.h
config.status: creating include/apu.h
config.status: creating include/apu_want.h
config.status: creating test/Makefile
config.status: creating include/private/apu_config.h
config.status: executing default commands
```

Figura 3.43 Salida de la configuración del paquete apr-util.


Figura 3.44 Compilación del paquete apr-util.

```
gcc -E -DHAVE_CONFIG_H  -DLINUX -D_REENTRANT -D_GNU_SOURCE   -I/sw/apr-util-1.5.4/include -I/sw/apr-ut
il-1.5.4/include/private  -I/usr/local/apr/include/apr-1  -I/sw/apr-util-1.5.4/xml/expat/lib  exports.
c | grep "ap_hack_" | sed -e 's/^.*[)]\(.*\);$/\1/' >> aprutil.exp
gcc -E -DHAVE_CONFIG_H  -DLINUX -D_REENTRANT -D_GNU_SOURCE   -I/sw/apr-util-1.5.4/include -I/sw/apr-ut
il-1.5.4/include/private  -I/usr/local/apr/include/apr-1  -I/sw/apr-util-1.5.4/xml/expat/lib  export_v
ars.c | sed -e 's/^\#[^!]*//' | sed -e '/^$/d' >> aprutil.exp
sed 's,^\(location=\).*$,\1installed,' < apu-1-config > apu-config.out
make[1]: Leaving directory `/sw/apr-util-1.5.4'
```

Figura 3.45 Salida de la compilación del paquete apr-util.

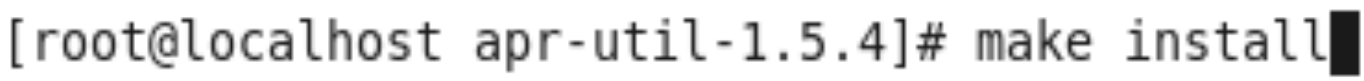



Figura 3.46 Instalación del paquete apr-util.

```
See any operating system documentation about shared libraries for
more information, such as the ld(1) and ld.so(8) manual pages.
----------------------------------------------------------------------
/usr/bin/install -c -m 644 aprutil.exp /usr/local/apr/lib
/usr/bin/install -c -m 755 apu-config.out /usr/local/apr/bin/apu-1-config
```

Figura 3.47 Salida de la instalación del paquete apr-util.

Una vez terminados los pasos anteriores salir de la carpeta con el comando **cd ..** , y se va a seguir con el paquete de pcre, para el cual serán pasos similares que con apr, sólo

que en este caso se configurará, compilará e instalará. En la figura 3.48 se mostrará el comando para descomprimir el paquete de pcre.

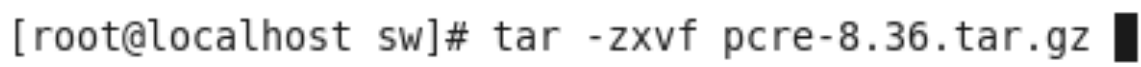



Figura 3.48 Descompresión del paquete pcre.

En las figuras 3.49, 3.50, 3.51, 3.52, 3.53 y 3.54 se mostrarán los pasos a seguir para la instalación así como las salidas de los comandos utilizados en las siguientes figuras.

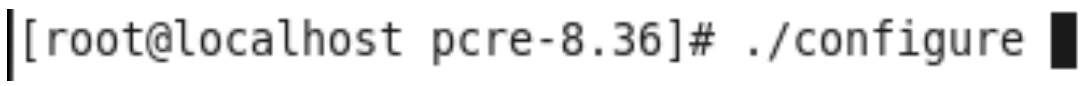



Figura 3.49 Configuración del paquete pcre.

```
\R matches only ANYCRLF ......... : no
EBCDIC coding .................... : no
EBCDIC code for NL ............... : n/a
Rebuild char tables .............. : no
Use stack recursion .............. : yes
POSIX mem threshold .............. : 10
Internal link size ............... : 2
Nested parentheses limit ........ : 250
Match limit ...................... : 10000000
Match limit recursion ............ : MATCH_LIMIT
Build shared libs ................ : yes
Build static libs ................ : yes
Use JIT in pcregrep .............. : no
Buffer size for pcregrep ........ : 20480
Link pcregrep with libz .......... : no
Link pcregrep with libbz2 ........ : no
Link pcretest with libedit ....... : no
Link pcretest with libreadline .. : no
Valgrind support ................. : no
Code coverage .................... : no
```

Figura 3.50 Salida de la configuración del paquete pcre.

```
[root@localhost pcre-8.36]# make
```

Figura 3.51 Compilación del paquete pcre.

```
CC       libpcre_la-pcre_xclass.lo
CC       libpcre_la-pcre_chartables.lo
CCLD     libpcre.la
CC       libpcreposix_la-pcreposix.lo
CCLD     libpcreposix.la
CXX      libpcrecpp_la-pcrecpp.lo
CXX      libpcrecpp_la-pcre_scanner.lo
CXX      libpcrecpp_la-pcre_stringpiece.lo
CXXLD    libpcrecpp.la
CC       pcretest-pcretest.o
CC       pcretest-pcre_printint.o
CCLD     pcretest
CC       pcregrep-pcregrep.o
CCLD     pcregrep
CXX      pcrecpp_unittest-pcrecpp_unittest.o
CXXLD    pcrecpp_unittest
CXX      pcre_scanner_unittest-pcre_scanner_unittest.o
CXXLD    pcre_scanner_unittest
CXX      pcre_stringpiece_unittest-pcre_stringpiece_unittest.o
CXXLD    pcre_stringpiece_unittest
```

Figura 3.52 Salida de la compilación del paquete pcre.

```
[root@localhost pcre-8.36]# make install
```

Figura 3.53 Instalación del paquete pcre.

```
ln -sf pcre_free_study.3                /usr/local/share/man/man3/pcre32_free_study.3
ln -sf pcre_free_substring.3            /usr/local/share/man/man3/pcre32_free_substring.3
ln -sf pcre_free_substring_list.3       /usr/local/share/man/man3/pcre32_free_substring_list.3
ln -sf pcre_fullinfo.3                  /usr/local/share/man/man3/pcre32_fullinfo.3
ln -sf pcre_get_named_substring.3       /usr/local/share/man/man3/pcre32_get_named_substring.3
ln -sf pcre_get_stringnumber.3          /usr/local/share/man/man3/pcre32_get_stringnumber.3
ln -sf pcre_get_stringtable_entries.3   /usr/local/share/man/man3/pcre32_get_stringtable_entries.3
ln -sf pcre_get_substring.3             /usr/local/share/man/man3/pcre32_get_substring.3
ln -sf pcre_get_substring_list.3        /usr/local/share/man/man3/pcre32_get_substring_list.3
ln -sf pcre_jit_exec.3                  /usr/local/share/man/man3/pcre32_jit_exec.3
ln -sf pcre_jit_stack_alloc.3           /usr/local/share/man/man3/pcre32_jit_stack_alloc.3
ln -sf pcre_jit_stack_free.3            /usr/local/share/man/man3/pcre32_jit_stack_free.3
ln -sf pcre_maketables.3                /usr/local/share/man/man3/pcre32_maketables.3
ln -sf pcre_pattern_to_host_byte_order.3 /usr/local/share/man/man3/pcre32_pattern_to_host_byte_order
ln -sf pcre_refcount.3                  /usr/local/share/man/man3/pcre32_refcount.3
ln -sf pcre_study.3                     /usr/local/share/man/man3/pcre32_study.3
ln -sf pcre_utf32_to_host_byte_order.3  /usr/local/share/man/man3/pcre32_utf32_to_host_byte_order.3
ln -sf pcre_version.3                   /usr/local/share/man/man3/pcre32_version.3
make[3]: Leaving directory `/sw/pcre-8.36'
make[2]: Leaving directory `/sw/pcre-8.36'
make[1]: Leaving directory `/sw/pcre-8.36'
```

Figura 3.54 Salida de la instalación del paquete pcre.

Al igual que con los demás paquetes hay que salir de la carpeta con el comando **cd ..** , con esto último ya se tendrán instalados paquetes apr, apr-util y pcre, solo falta instalar el servicio web apache, para esto se van a realizar pasos similares, pero a diferencia de los demás que se instalan de manera automática dentro del sistema, el servicio apache se va ha a colocar en una carpeta en específico, esto ayudará a encontrar más rápido las configuraciones y para mayor seguridad de nuestro servicio, se va necesitar crear una carpeta llamada servicio, en la figura 3.55 se mostrará el comando que se utilizará para la creación de la carpeta que al igual que la carpeta sw que creamos va a estar dentro de la raíz del sistema.


Figura 3.55 Creación de la carpeta servicios.

Para iniciar con la instalación hay que descomprimir el paquete de los servicios con el comando tar que se verá en la figura 3.56, entrar a la carpeta con cd espacio y el nombre de la carpeta. Después se va continuar con la configuración del paquete de apache, aquí se modificará el comando con la opción - -prefix= y le agregaremos la ruta de la carpeta recién creada más el nombre de la carpeta que va a tener nuestro servicio, para este caso va a ser _apache, en las figuras 3.57, 3.58, 3.59, 3.60, 3.61, y 3.62.

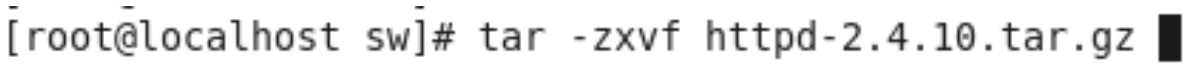



Figura 3.56 Descompresión del paquete httpd.

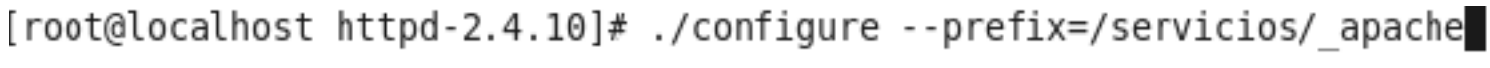



Figura 3.57 Configuración del paquete httpd.

```
config.status: creating docs/conf/extra/httpd-manual.conf
config.status: creating docs/conf/extra/httpd-mpm.conf
config.status: creating docs/conf/extra/httpd-multilang-errordoc.conf
config.status: creating docs/conf/extra/httpd-ssl.conf
config.status: creating docs/conf/extra/httpd-userdir.conf
config.status: creating docs/conf/extra/httpd-vhosts.conf
config.status: creating docs/conf/extra/proxy-html.conf
config.status: creating include/ap_config_layout.h
config.status: creating support/apxs
config.status: creating support/apachectl
config.status: creating support/dbmmanage
config.status: creating support/envvars-std
config.status: creating support/log_server_status
config.status: creating support/logresolve.pl
config.status: creating support/phf_abuse_log.cgi
config.status: creating support/split-logfile
config.status: creating build/rules.mk
config.status: creating build/pkg/pkginfo
config.status: creating build/config_vars.sh
config.status: creating include/ap_config_auto.h
config.status: executing default commands
```

Figura 3.58 Salida de la configuración del paquete httpd.

```
[root@localhost httpd-2.4.10]# make
```

Figura 3.59 Compilación del paquete httpd.

```
-pic -c mod_alias.c && touch mod_alias.slo
/usr/local/apr/build-1/libtool --silent --mode=link gcc -std=gnu99  -g -O2 -pthread           -o mod_a
lias.la -rpath /servicios/_apache/modules -module -avoid-version  mod_alias.lo
/usr/local/apr/build-1/libtool --silent --mode=compile gcc -std=gnu99  -g -O2 -pthread      -DLINUX -D
_REENTRANT -D_GNU_SOURCE     -I. -I/sw/httpd-2.4.10/os/unix -I/sw/httpd-2.4.10/include -I/usr/local/ap
r/include/apr-1 -I/usr/local/include -I/sw/httpd-2.4.10/modules/aaa -I/sw/httpd-2.4.10/modules/cache -
I/sw/httpd-2.4.10/modules/core -I/sw/httpd-2.4.10/modules/database -I/sw/httpd-2.4.10/modules/filters
-I/sw/httpd-2.4.10/modules/ldap -I/sw/httpd-2.4.10/modules/loggers -I/sw/httpd-2.4.10/modules/lua -I/s
w/httpd-2.4.10/modules/proxy -I/sw/httpd-2.4.10/modules/session -I/sw/httpd-2.4.10/modules/ssl -I/sw/h
ttpd-2.4.10/modules/test -I/sw/httpd-2.4.10/server -I/sw/httpd-2.4.10/modules/arch/unix -I/sw/httpd-2.
4.10/modules/dav/main -I/sw/httpd-2.4.10/modules/generators -I/sw/httpd-2.4.10/modules/mappers -prefer
-pic -c mod_rewrite.c && touch mod_rewrite.slo
/usr/local/apr/build-1/libtool --silent --mode=link gcc -std=gnu99  -g -O2 -pthread           -o mod_r
ewrite.la -rpath /servicios/_apache/modules -module -avoid-version  mod_rewrite.lo
make[4]: Leaving directory `/sw/httpd-2.4.10/modules/mappers'
make[3]: Leaving directory `/sw/httpd-2.4.10/modules/mappers'
make[2]: Leaving directory `/sw/httpd-2.4.10/modules'
make[2]: Entering directory `/sw/httpd-2.4.10/support'
make[2]: Leaving directory `/sw/httpd-2.4.10/support'

make[1]: Leaving directory `/sw/httpd-2.4.10'
```

Figura 3.60 Salida de la compilación del paquete httpd.

```
[root@localhost httpd-2.4.10]# make install
```

Figura 3.61 Instalación del paquete httpd.

```
mkdir /servicios/_apache/conf/original
mkdir /servicios/_apache/conf/original/extra
Installing HTML documents
mkdir /servicios/_apache/htdocs
Installing error documents
mkdir /servicios/_apache/error
Installing icons
mkdir /servicios/_apache/icons
mkdir /servicios/_apache/logs
Installing CGIs
mkdir /servicios/_apache/cgi-bin
Installing header files
mkdir /servicios/_apache/include
Installing build system files
mkdir /servicios/_apache/build
Installing man pages and online manual
mkdir /servicios/_apache/man
mkdir /servicios/_apache/man/man1
mkdir /servicios/_apache/man/man8
mkdir /servicios/_apache/manual
make[1]: Leaving directory `/sw/httpd-2.4.10'
```

Figura 3.62 Salida de la instalación del paquete httpd.

Para finalizar la instalación copiar el demonio el cual se encargará de activarlo cuando se ejecute, así como detenerlo o reiniciarlo. En la figura 3.63 se mostrará la forma en la que se realizará la copia, para empezar con el servicio se tendrá que ejecutar el comando service httpd start, con esto el servicio de apache estará activado y listo para la configuración. En el caso de una migración como esta se tendrá que verificar si existe alguna configuración dentro del archivo en la carpeta apache/conf/ , el archivo llamado httpd.conf tendrá las configuraciones, en el caso de esta migración no se tiene ninguna modificación necesaria hasta este momento.

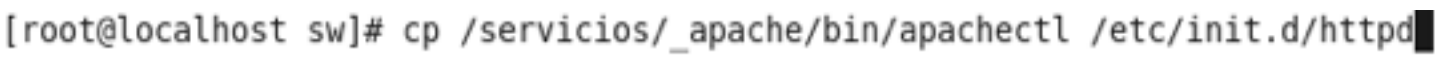



Figura 3.63 Copia del demonio de *apache*.

### 3.2.2 MYSQL

Como se mencionó al inicio de esta sección el servicio mysql es necesario para la plataforma *moodle.* Este servicio es necesario para el almacenamiento de los datos de los alumnos, profesores y otros académicos que usan esta plataforma. Una vez que ya se tiene el respaldo correspondiente de la base de datos que se tiene en el servidor actual, se necesita instalar este servicio en cada uno de los nodos y con el respaldo dejar activo el servicio de mysql para que lo ocupe *moodle*. En la figura 3.64 se mostrará el comando **wget**, que al igual que con apache se descargara lo necesario para la instalación, en este caso sólo se necesita un paquete el cual se puede conseguir de la página de mysql, el paquete que está por ocuparse es para una arquitectura de 64 bits[51], para continuar se tendrá que descomprimir el archivo como se verá en la figura 3.65, se moverá la carpeta a la ruta /servicios/_mysql como se muestra en la figura 3.66, se necesitará crear un usuario y un grupo específico, el cual será mysql como usuario y mysql como grupo en la figura 3.67 se muestra cómo se realizará la creación de estos.

Una vez que se tiene el grupo y el usuario, entrar a la carpeta /servicios/_mysql con el comando cd espacio y la ruta de la carpeta, una vez dentro modificar la pertenencia de la carpeta, ya que el usuario y grupo que tienen privilegios en la carpeta es el usuario administrador root, por eso se cambiarán los permisos de propiedad en la

[51] En caso de necesitar el paquete para otra arquitectura o para otro sistema operativo, se debe ir a la página de mysql y buscar en la parte de abajo mysql server y elegir el sistema operativo y la arquitectura que se necesite.

figura 3.68 se muestra el comando a utilizar, así como en la figura 3.69 la salida de éste.

```
[root@prueba1 sw]# wget http://dev.mysql.com/get/Downloads/MySQL-5.6/mysql-5.6.22
2-linux-glibc2.5-x86_64.tar.gz_
```

Figura 3.64 Descarga del paquete mysql.

```
[root@prueba1 sw]# tar -zxvf mysql-5.6.22-linux-glibc2.5-x86_64.tar.gz_
```

Figura 3.65 Descompresión del paquete mysql.

```
[root@prueba1 sw]# mv mysql-5.6.22-linux-glibc2.5-x86_64 /servicios/_mysql_
```

Figura 3.66 Comando **mv** para mover la carpeta mysql a la ruta de la carpeta servicios.

```
[root@prueba1 sw]# groupadd mysql
[root@prueba1 sw]# useradd -r -g mysql mysql
```

Figura 3.67 Creación del grupo y del usuario mysql.


Figura 3.68 Cambio de propiedad de la carpeta mysql.

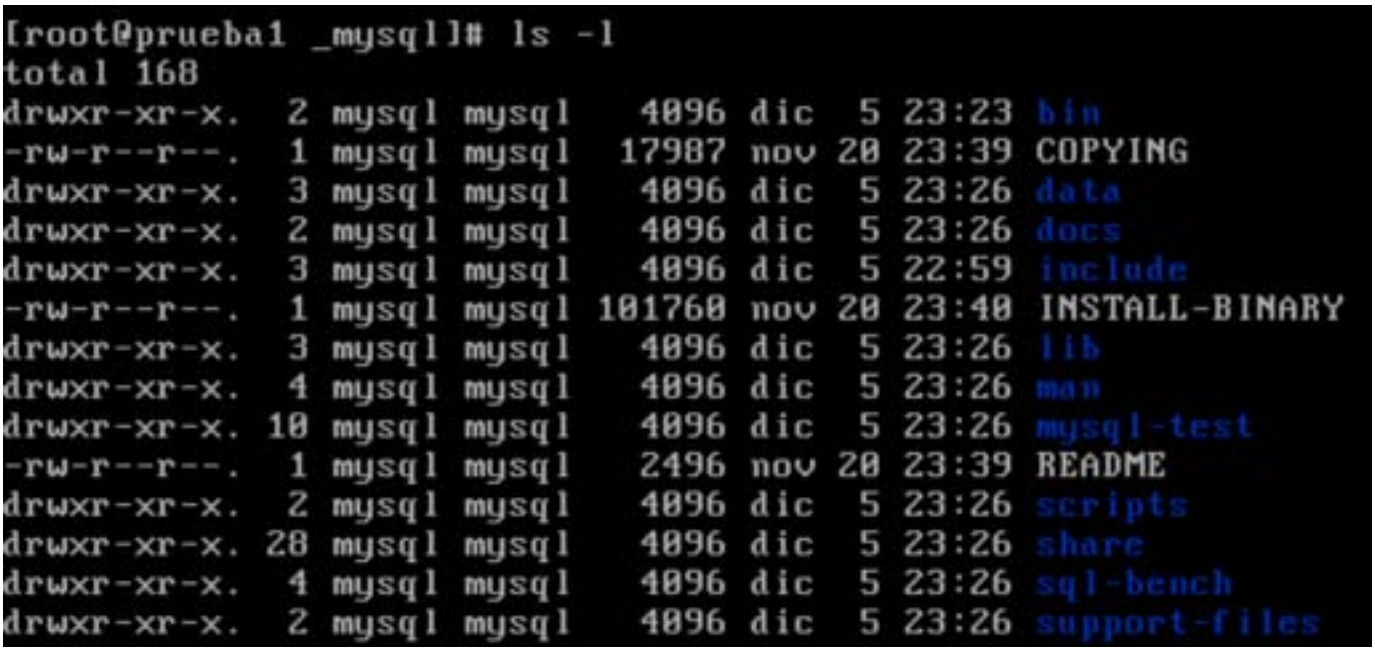



Figura 3.69 Comprobación de propiedad en la carpeta mysql.

Una vez hecho esto se va a comenzar con la instalación del servicio de base de datos mysql, para esto se va a comenzar ejecutando el script que viene en la figura 3.70, y en la figura 3.71 se observara la salida de éste.


Figura 3.70 Script de instalación de mysql.

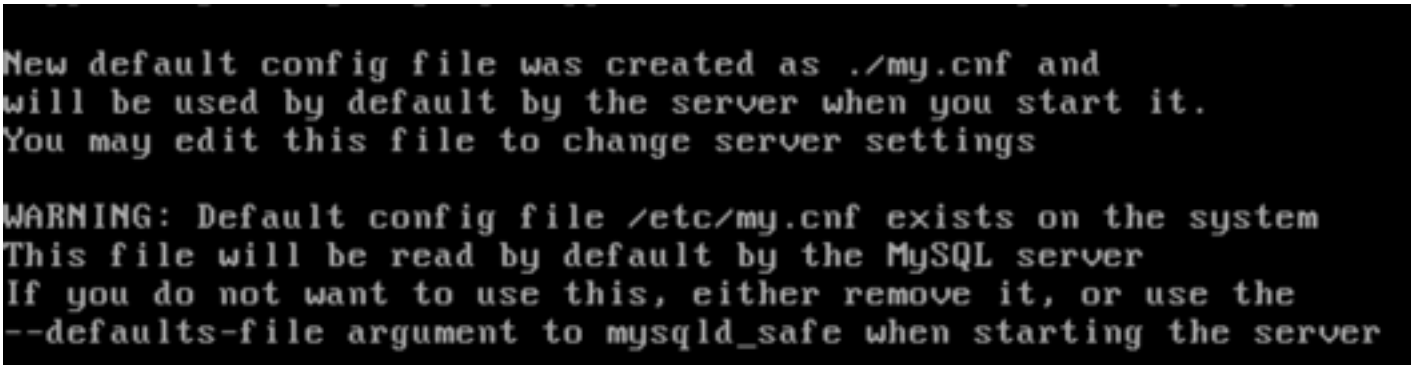



Figura 3.71 Salida de la ejecución del script de instalación.

Lo siguiente a realizar será volver a cambiar la pertenencia de la carpeta a *root*, pero el grupo seguir siendo de mysql, la carpeta **data** deberá seguir siendo de mysql, en las figuras 3.72 y 3.73 se mostrará como se realizan los cambios.


Figura 3.72 Cambio de propiedad a root de la carpeta mysql.


Figura 3.73 Cambio de propiedad a mysql de la carpeta data dentro de mysql.

Una vez realizados estos cambios se tendrán que mover el archivo de configuración a la carpeta etc, para eso en la figura 3.74 se muestra el comando y la ruta del archivo de configuración. Para iniciar el servicio en modo seguro para ver si la instalación fue correcta se va ejecutar el demonio mysqld_safe el cual se muestra en la figura 3.75, una vez que se haya visto que no se generó ningún error copiar el demonio de ejecución del servicio a la ruta /etc/init.d con el nombre de mysqld como se muestra en la figura 3.76 y se le darán permisos para que el servicio se active durante el inicio del sistema, en la figura 3.77 se muestra el comando a seguir.

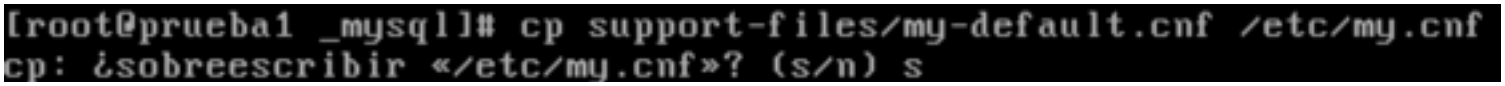



Figura 3.74 Copia del archivo de configuración de mysql.


Figura 3.75 Ejecución del demonio mysqld_safe.


Figura 3.76 Copia del demonio mysqld.


Figura 3.77 Comando chkconfig que dará permisos de ejecución al arranque del sistema.

Para la configuración final editar el archivo my.cnf que se copió a la carpeta /etc, así como también al demonio mysql que colocado en la carpeta /etc/init.d, en los dos modificaremos las mismás donde se especificará donde se encuentran nuestra carpeta de mysql y la carpeta **data**, en las figuras 3.78, 3.79 y 3.80 se muestra lo anteriormente explicado.


Figura 3.78 Edición del archivo my.cnf.


Figura 3.79 Edición del demonio mysqld.

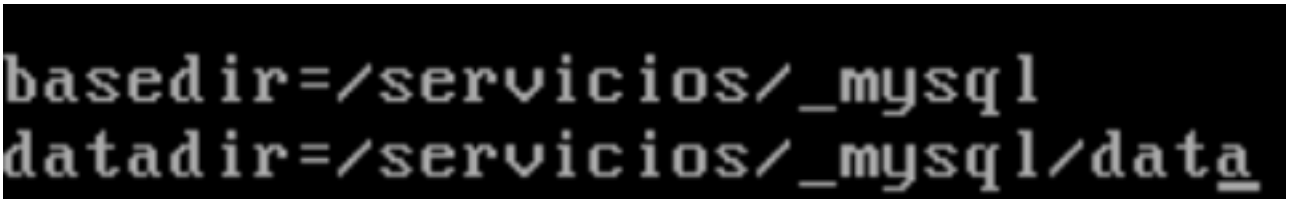



Figura 3.80 Líneas a editar en el archivo my.cnf y demonio mysqld.

Una vez realizados los siguientes cambios reiniciar el servicio con el comando service mysqld restart, con esto el servicio está listo para funcionar, pero aún no se puede trabajar desde línea de comandos, para hacer esto modificar el archivo /etc/ profile, como se ve en las figuras 3.81, 3.82 y 3.83, en la figura 3.82 las líneas que se editarán se deben poner al final del archivo para que no afecte ningúna configuración del sistema.


Figura 3.81 Edición del archivo /etc/profile.

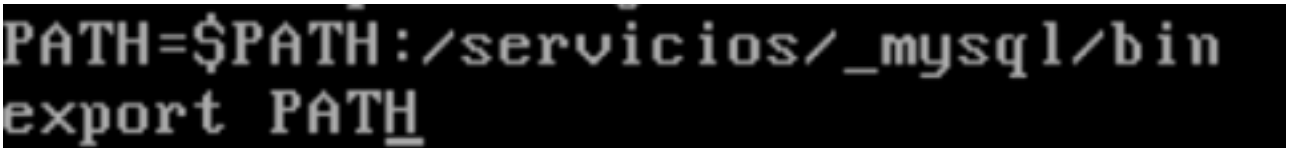



Figura 3.82 Líneas necesarias para la ejecución del servicio mysql en línea de comandos.


Figura 3.83 Comando para cargar las nuevas configuraciones.

Una vez realizado lo anterior cada que se ejecute el comando mysql se podrá trabajar con él desde la línea de comandos como se muestra en la figura 3.84.

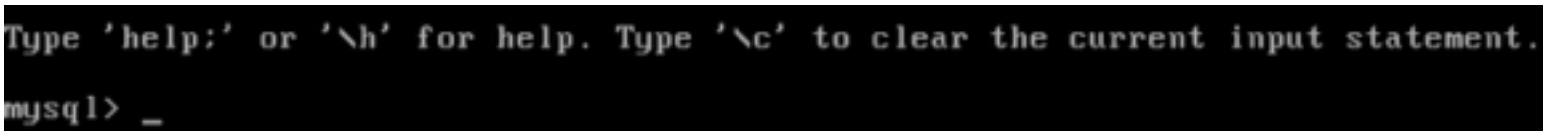



Figura 3.84 Ejecución del comando mysql.

Para el caso de la migración se debe de tener el respaldo en el servidor que se vaya a migrar, se utilizará el comando mysql -u "usuario administrador" -p < "nombre del respaldo".sql[52], con esto el respaldo de la base de datos de *moodle* se cargará en el servidor teniendo la misma información que en original.

### 3.2.3 PHP

PHP es un lenguaje de programación orientado a la parte de servicios web, este servicio es necesario, ya que la plataforma moodle tiene todos sus recursos en éste, por este motivo es indispensable su instalación, para esto al igual que los demás paquetes se descargará con **wget** como se muestra en la figura 3.85. El archivo que se descargará es un index.html al cual se le deberá cambiar al nombre por php.tar.gz, se

[52] La opción -u en el comando se refiere al usuario y la opción -p hace referencia al password del usuario antes mencionado.

tendrá que descomprimir con el comando tar que se utilizan con los demás paquetes para ejecutar los archivos de configuración, compilación e instalación, para la instalación requerida por *moodle* se debe de descargar ciertas librerías las cuales se muestran en la figura 3.86 junto con el comando yum.


Figura 3.85 Descarga del paquete de php.


Figura 3.86 Descarga de librerías con el comando yum.

Entrar a la carpeta con **cd** espacio, el nombre de la carpeta, y ejecutar el siguiente comando:

/configure —with-mysql=/servicios/_mysql —with-apx2=/ servicios/_apache/ bin/apxs —prefix=/servicios/_php --with-config-file-path=/etc/ --with-curl --with-xmlrpc --enable-mbstring --with- mysqli=/servicios/_mysql/bin/mysql_config” --enable-soap --with-zlib- dir=ext/zlib” --enable-zip --enable-intl —with-jpeg-dir=/usr/share/doc/libjpeg-turbo-1.2.1 —with-png-dir=/usr/share/doc/libpng-1.2.49 --with- xpm-dir=/usr/ share/doc/libXpm-3.5.10 —with-freetype-dir=/usr/share/doc/ freetype-2.3.11 —with-gd

En la figura 3.87 se muestra el comando anteriormente puesto y en la figura 3.88 la salida de este, una vez terminada la configuración en las figuras 3.89, 3.90, 3.91 y 3.92 se muestra la compilación y la instalación con sus respectivas salidas.

```
[root@prueba1 php-5.6.5]# ./configure --with-mysql=/servicios/_mysql/ --with-apx
s2=/servicios/_apache/bin/apxs --prefix=/servicios/_php --with-config-file-path=
/etc --with-curl --with-xmlrpc --enable-mbstring --with-mysqli=/servicios/_mysql
/bin/mysql_config --enable-soap --with-zlib-dir=ext/zlib/ --enable-zip --enable-
intl --with-jpeg-dir=/usr/share/doc/libjpeg-turbo-1.2.1/ --with-png-dir=/usr/sha
re/doc/libpng-1.2.49/ --with-xpm-dir=/usr/share/doc/libXpm-3.5.10/ --with-freety
pe-dir=/usr/share/doc/freetype-2.3.11/ --with-gd_
```

Figura 3.87 Comando de configuración de php.

```
config.status: creating php5.spec
config.status: creating main/build-defs.h
config.status: creating scripts/phpize
config.status: creating scripts/man1/phpize.1
config.status: creating scripts/php-config
config.status: creating scripts/man1/php-config.1
config.status: creating sapi/cli/php.1
config.status: creating sapi/cgi/php-cgi.1
config.status: creating ext/phar/phar.1
config.status: creating ext/phar/phar.phar.1
config.status: creating main/php_config.h
config.status: executing default commands
```

Figura 3.88 Salida del comando de configuración


Figura 3.89 Compilación del servicio php.

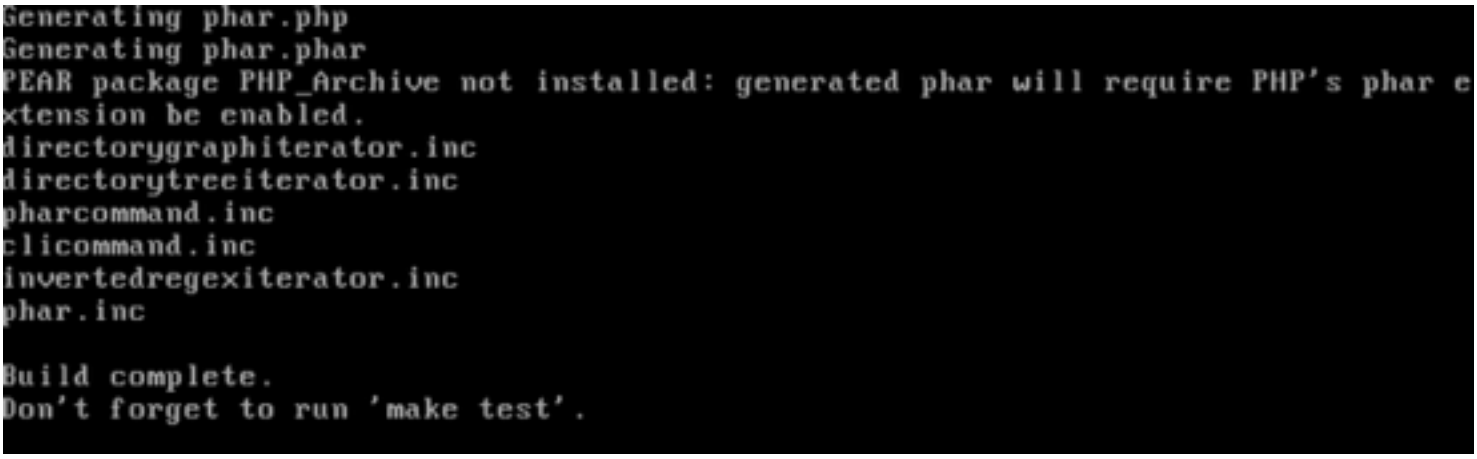



Figura 3.90 Salida de la compilación.


Figura 3.91 Instalación de php.


Figura 3.92 Salida de la instalación.

Ya lista la instalación se va a realizar una serie de configuraciones que ayudarán a que el lenguaje de programación php trabaje con el servicio web *apache*, para esto se deben de configurar unas líneas dentro del archivo httpd.conf que se encuentra en la carpeta de *apache*, en la figura 3.93 y 3.94 se muestra la ruta del archivo, así como las líneas que se deben de agregar, en la figura 3.95 se modifica otra línea que es necesaria para la configuración de php y de la plataforma *moodle*.


Figura 3.93 Edición del archivo httpd.conf


Figura 3.94 Líneas que se deben de agregar para la comunicación entre *apache* y php.

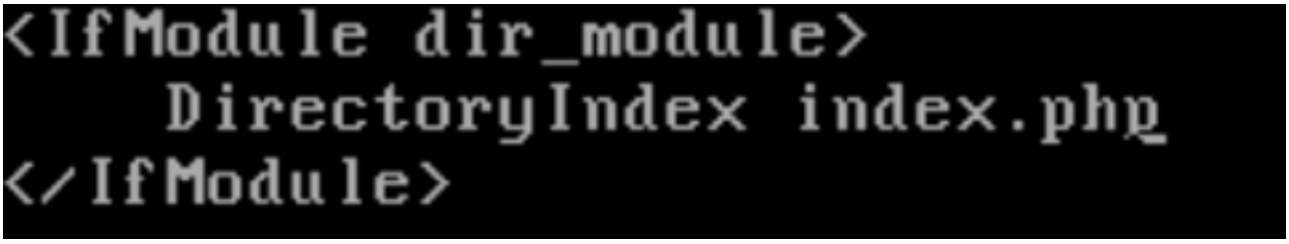
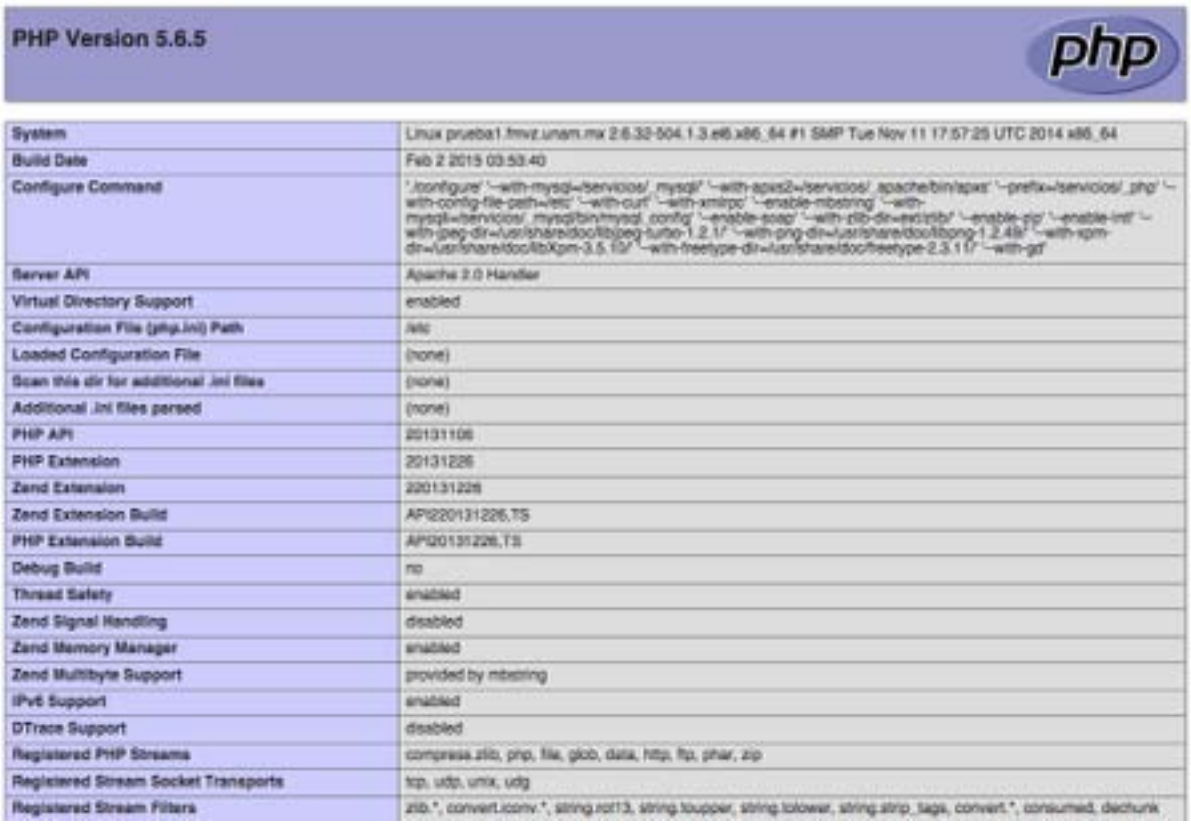



Figura 3.95 Cambio de línea índex.html a índex.php.

Ya que se agregaron las líneas de la figura 3.94 y la modificación de la línea 3.95, guardar el archivo saliendo del modo de edición con ESC seguido de :wq!, después crear un archivo llamado phpinfo.php en la carpeta de apache dentro de htdocs, en la figura 3.96 se muestra la forma de crear el archivo y en la figura 3.97 se muestra el contenido, después de realizar estos cambios ir al navegador web y colocar la dirección ip del servidor, saldrá una pantalla en blanco junto con dos links en color azul, uno de ellos es el archivo que se acaba de crear si toda la instalación fue correcta aparecerá algo similar a la figura 3.98, con esto se verifica que php se encuentra instalado con las librerías que se necesitan.


Figura 3.96 Creación del archivo phpinfo.php


Figura 3.97 Contenido del archivo phpinfo.php



**PHP Version 5.6.5**

| | |
|---|---|
| System | Linux prueba1.fmvz.unam.mx 2.6.32-504.1.3.el6.x86_64 #1 SMP Tue Nov 11 17:57:25 UTC 2014 x86_64 |
| Build Date | Feb 2 2015 03:53:40 |
| Configure Command | './configure' '--with-mysql=/servicios/_mysql' '--with-apxs2=/servicios/_apache/bin/apxs' '--prefix=/servicios/_php' '--with-config-file-path=/etc' '--with-curl' '--with-xmlrpc' '--enable-mbstring' '--with-mysqli=/servicios/_mysql/bin/mysql_config' '--enable-soap' '--with-zlib-dir=/ext/zlib/' '--enable-zip' '--enable-intl' '--with-jpeg-dir=/usr/share/doc/libjpeg-turbo-1.2.1/' '--with-png-dir=/usr/share/doc/libpng-1.2.49/' '--with-xpm-dir=/usr/share/doc/libXpm-3.5.10/' '--with-freetype-dir=/usr/share/doc/freetype-2.3.11/' '--with-gd' |
| Server API | Apache 2.0 Handler |
| Virtual Directory Support | enabled |
| Configuration File (php.ini) Path | /etc |
| Loaded Configuration File | (none) |
| Scan this dir for additional .ini files | (none) |
| Additional .ini files parsed | (none) |
| PHP API | 20131106 |
| PHP Extension | 20131226 |
| Zend Extension | 220131226 |
| Zend Extension Build | API220131226,TS |
| PHP Extension Build | API20131226,TS |
| Debug Build | no |
| Thread Safety | enabled |
| Zend Signal Handling | disabled |
| Zend Memory Manager | enabled |
| Zend Multibyte Support | provided by mbstring |
| IPv6 Support | enabled |
| DTrace Support | disabled |
| Registered PHP Streams | compress.zlib, php, file, glob, data, http, ftp, phar, zip |
| Registered Stream Socket Transports | tcp, udp, unix, udg |
| Registered Stream Filters | zlib.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk |

Figura 3.98 Comprobación de instalación de php.

### 3.2.4 MOODLE

Como se había mencionado anteriormente *moodle* es una plataforma web, con múltiples usos en el ámbito académico, por lo cual la carga de trabajo va a ir creciendo con el paso de los años, por esta razón es indispensable la actualización de los servicios. En el caso de la migración de esta plataforma es bastante simple, se tiene que migrar la base de datos a una nueva, lo cual se hizo al final del punto 3.2.2 y se tiene que migrar donde esta plataforma almacena toda la información de los cursos, exámenes, etc. El cual por lo regular se llama moodledata[53]; esta carpeta contiene los exámenes, cursos y otras cosas que se puedan agregar. La carpeta debe de tener permisos de lectura, escritura y ejecución, para la migración la carpeta deberá encontrarse en la misma ruta que en servidor donde se tiene actualmente, esto por que *moodle* guarda esta información en una archivo llamado config.php, donde también viene el nombre de la base de datos del servidor y otras cosas importantes para la plataforma, si se altera la ruta y los permisos podría generar problemas si no se modifica en el config.php. Por último el config.php se tiene que llevar a la nueva carpeta donde *moodle* guarde sus archivos de configuración; para esto primero se tendrá que descargar de la página oficial de moodle el archivo .tgz, después se descomprimirá de la misma forma que los demás, en la figura 3.99 se mostrará la descompresión del archivo. Continuando con la migración se moverá la carpeta moodle a la carpeta servicios lo cual se mostrará en la figura 3.100 y se pondrá el config.php adentro y la carpeta de moodledata en la ruta especificada o en una nueva pero sin olvidar la modificación del archivo de configuración.

[53] Esta carpeta de puede llamar de otra forma, para saber su nombre y ruta exacta checar el config.php delntro de la carpeta de moodle que se tenga instalado.

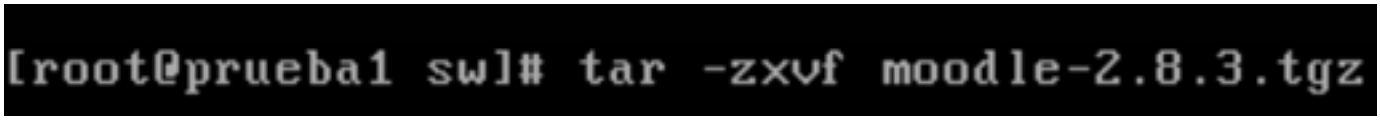



Figura 3.99 Descompresión del paquete moodle.


Figura 3.100 Comando mv para mover la carpeta moodle a la ruta /servicios.

Una vez que se haya terminado de mover la carpeta *moodle*, se haya puesto adentro el config.php y que la carpeta de moodledata se encuentra en la ruta especificada, lo único que queda por configurar es el servicio web apache. Hay que dirigirse al archivo de configuración de apache el httpd.config, donde tse tendrá que colocar la ruta de nuestro *moodle* para que lea los archivos .php y la página quede disponible para el usuario, en las figuras 3.101 y 3.102 se muestra lo anterior.


Figura 3.101 Línea a editar del httpd.config.

```
DocumentRoot "/servicios/moodle"
<Directory "/servicios/moodle">
```

Figura 3.102 Línea editada con la ruta de la carpeta *moodle.*

Una vez modificadas estas líneas, guardar el archivo saliendo del modo de edición con ESC seguido de :wq!, reiniciar el servicio web con el comando service httpd restart, cuando el servicio vuelva a iniciar iremos al navegador y colocar la ip del servidor actual y el resultado se verá en la figura 3.103.


Figura 3.103 Resultados de la migración del servidor de Apuntes en línea.

## 3.3 RESULTADOS DE LA IMPLEMENTACIÓN

Después de que se haya migrado la plataforma moodle a cada uno de los nodos que van a estar dentro del balanceador de carga, se va a volver a medir el rendimiento del servidor como se hizo anteriormente para llegar a medir si llegó al objetivo que era mejorar el número de solicitudes y el tiempo o igual el número de solicitudes pero con tiempos más satisfactorios, para lo siguiente se volverá a ocupar el comando ab, se tendrá que volver a instalar pero ahora en la máquina de balanceo de carga como se hizo en el capítulo anterior.

### 3.3.1 PRUEBAS DE RENDIMIENTO

Se van a realizar dos pruebas en esta ocasión ya que fueron requeridas por el profesor German Valero, a continuación se mostrará los resultados del servidor de balanceo de carga y alta disponibilidad[54], las pruebas se realizarán con el comando ab anteriormente usado.

Se comenzará la prueba con 100 conexiones y 50 al mismo tiempo para analizar los tiempos que frecuentemente se podrían llegar a tener, cada que un grupo inicie sesión dentro del balanceador de carga.

Mediciones del rendimiento del 132.248.62.33

Con 100 conexiones y 50 al mismo tiempo:

ab -n 100 -c 50 132.248.62.33/index.php
This is ApacheBench, Version 2.3 <$Revision: 655654 $>

[54] Los siguientes resultados provienen del documento enviado al profesor Valero y que fueron aprobados por él.

Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 132.248.62.33 (be patient).....done

Server Software: Apache/2.4.3

Server Hostname: 132.248.62.33

Server Port: 80

Document Path: /index.php

Document Length: 207 bytes

Concurrency Level: 50

Time taken for tests: 0.023 seconds

Complete requests: 100

Failed requests: 0

Write errors: 0

Non-2xx responses: 100

Total transferred: 39400 bytes

HTML transferred: 20700 bytes

Requests per second: 4267.49 [#/sec] (mean)

Time per request: 11.716 [ms] (mean)

Time per request: 0.234 [ms] (mean, across all concurrent requests)

Transfer rate: 1641.98 [Kbytes/sec] received

Connection Times (ms)

| | min | mean[+/-sd] | median | max |
|---|---|---|---|---|
| Connect: | 2 | 5 1.9 | 5 | 9 |
| Processing: | 2 | 5 2.0 | 5 | 10 |
| Waiting: | 0 | 4 2.5 | 4 | 9 |
| Total: | 9 | 11 1.1 | 11 | 13 |

Percentage of the requests served within a certain time (ms)

50% 11

66% 12

75% 12

80% 12

90% 12

95% 12

98% 12

99% 13

100% 13 (longest request)

Como se puede notar los tiempos de reacción que pueden llegar a tener son mínimos , esto habla de que el servidor de balanceo de carga y alta disponibilidad no tiene problema alguno con las conexiones que frecuentemente llega a tener el servidor actual, pero como el punto de este sistema es llegar a medir el máximo número de conexiones, como se estableció en la parte del balanceo de carga la máxima cantidad de conexiones que nosotros establecimos fue de 600[55], por lo que si mandamos 1000 conexiones y 500 al mismo tiempo no debería de ocurrir ningún problema, a continuación se mostrará los resultados:

Para 1000 conexiones y 500 al mismo tiempo:

ab -n 1000 -c 500 132.248.62.33/index.php

This is ApacheBench, Version 2.3 <$Revision: 655654 $>

Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

Licensed to The Apache Software Foundation, http://www.apache.org/

[55] Nota: Este número fue arbitrario se puede aumentar o disminuir conforme a las capacidades de nuestro servidor de balanceo de carga y alta disponibilidad.

Benchmarking 132.248.62.33 (be patient)

Completed 100 requests

Completed 200 requests

Completed 300 requests

Completed 400 requests

Completed 500 requests

Completed 600 requests

Completed 700 requests

Completed 800 requests

Completed 900 requests

Completed 1000 requests

Finished 1000 requests

Server Software: Apache/2.4.3

Server Hostname: 132.248.62.26

Server Port: 80

Document Path: /index.php

Document Length: 207 bytes

Concurrency Level: 500

Time taken for tests: 0.248 seconds

Complete requests: 1000

Failed requests: 0

Write errors: 0

Non-2xx responses: 1074

Total transferred: 423156 bytes

HTML transferred: 222318 bytes

Requests per second: 4036.38 [#/sec] (mean)

Time per request: 123.874 [ms] (mean)

Time per request: 0.248 [ms] (mean, across all concurrent requests)

Transfer rate: 1667.99 [Kbytes/sec] received

Connection Times (ms)

| | min | mean[+/-sd] | | median | max |
|---|---|---|---|---|---|
| Connect: | 16 | 24 | 5.5 | 23 | 40 |
| Processing: | 9 | 42 | 55.8 | 26 | 226 |
| Waiting: | 0 | 33 | 58.3 | 17 | 224 |
| Total: | 41 | 66 | 54.4 | 50 | 245 |

Percentage of the requests served within a certain time (ms)

50% 50

66% 50

75% 55

80% 56

90% 57

95% 245

98% 245

99% 245

100% 245 (longest request)

Podemos observar que los tiempos suben de manera considerable, pero sin llegar a los extremos que se tuvieron en la prueba del servidor antes de la implementación del servidor de carga con alta disponibilidad, otro dato que es de vital importancia en los resultados en el número de solicitudes que no se pudieron llegar a concretar fue de 0, cuando el servidor antes del balanceo llegó a tener 171 solicitudes que no pudieron llegar a conectar con el servidor o devueltas al cliente, con esto, se puede llegar a decir que no sólo se bajaron los tiempos con los que el servidor recibe y devuelve la solicitud del recurso web que se necesita si no que también se pudo incrementar el número máximo de conexiones con tiempos excelentes de respuesta. Estas pruebas

demuestran que generar el servicio de balanceo de carga para 5 ó 6 servidores de igual tamaño o menores a él, pueden vivir en conjunto sin ningún problema y poder seguir teniendo tiempos de respuesta bastante satisfactorios con base a las necesidades de la Facultad de Medicina Veterinaria y Zootecnia.

### 3.3.2 CONSIDERACIONES

Dentro de este apartado se van a dar ciertas consideraciones necesarias para que no haya problemas en el uso de lo que anteriormente se instaló, se debe de tener en cuenta las reglas del *firewall*[56] local, ya que para la comunicación con los usuarios y entre los servidores debe de haber ciertas reglas en el iptables[57], las cuales son que tanto los nodos como el balanceador de carga deberán de tener abiertos el puerto 80[58] y el puerto 5000[59], así como también se debe de analizar que el modelo del balanceo de carga está basado en la arquitectura de la Facultad de Medicina Veterinaria y Zootecnia por lo que utilizarlo en otras instituciones puede no tener el mismo resultado. En el caso de la alta disponibilidad, esta viene en consideración al lugar donde se implemente, ya que en otro ambiente se deberá de modificar el modelo propuesto para un mejor servicio, así como instalar otros servicios que se crean pertinentes, en cuestión de la alta disponibilidad física se debe de contar con una planta de energía auxiliar y una buena topología en la red de la institución a la que se le vaya a instalar este servicio.

---

[56] Es una parte del sistema que bloquea, el acceso no autorizado permitiendo sólo las comunicaciones autorizadas.

[57] Sistema de *firewall* de centos.

[58] Puerto utilizado por el servicio apache para las solicitudes web.

[59] Puerto utilizado por haproxy para el balanceo y las comunicaciones con los nodos.

Para la instalación de los servicios en el apartado 3.2, de preferencia deben ser instalados por alguien responsable de los servidores o con conocimiento medio del sistema operativo linux.
En cuanto al config.php de los nodos, se le debe de quitar el nombre de la página de apuntes en línea, sólo se deberá dejar con la dirección ip, ya que nuestro balanceador de carga tendrá el dominio de apuntes en línea, al cual los usuarios solicitarán entrar.

Resumen del capítulo:

Al final de este capítulo se puede observar cómo se implementó el modelo propuesto en el capítulo 2 para resolver la problemática que se tiene con la página apuntes en línea, así como también se puede ver la instalación y la configuración del servicio *haproxy* para su funcionamiento como balanceador y generador de alta disponibilidad entre el y sus nodos. Se pudo mostrar una mejora sustancial del rendimiento de la página del profesor Valero y los posibles alcances que podría llegar a tener este modelo a largo plazo.
Se aprendió el modo de instalación y configuración de algunos servicios web como lo son *apache* y *mysql*, así como también la instalación de la plataforma *moodle* que es el punto central de la necesidad que se encuentra en la FMVZ.

# CONCLUSIONES

1. La herramienta *haproxy* es muy útil, ya que incluye no sólo al balanceo de carga si no que también genera alta disponibilidad entre los nodos, siendo muy útil para el ahorro de procesamiento de los equipos.
2. La implementación del balanceador de carga ayudó a disminuir los tiempos de una manera considerable; como lo vimos durante este trabajo, el balanceador bajó el tiempo de segundos a mili segundos, estos resultados superaron las expectativas que se necesitaban para que el proyecto fuera exitoso.
3. Mantener los servicios actualizados es una parte fundamental del rendimiento de los servicios web, ya que no solo pueden ser menos seguros, si no también sirve para mejora en el rendimiento y la velocidad.
4. En el caso de la implementación que se realizó no sólo los servicios que se instalaron se tienen que actualizar en su momento, también el balanceador se debe de actualizar y buscar parches por alguna posible afectación de personas dedicadas al ataque de los servicios web.
5. La Facultad de Ingeniería nos ayuda con los fundamentos para realizar la búsqueda de este tipo de soluciones que pueden llegar a ser de *software* libre o pueden ser de paga.

# BIBLIOGRAFÍA

http://redes-privadas-virtuales.blogspot.mx/2009/03/alta-disponibilidad-con-heartbeat.html

Marzo 8, 2009

Alta disponibilidad con heartbeat

https://es.wikipedia.org/wiki/Planificaci%C3%B3n_Round-robin

Mayo 22, 2015

Planificación Round Robin

http://lobobinario.blogspot.mx/2011/09/cluster-de-alta-disponibilidad-balanceo.html

Septiembre 1, 2011

Cluster de alta disponibilidad + balanceo de carga

http://www.slideshare.net/setoide/alta-disponibilidad-y-balanceo-de-carga-en-linux

Marzo 12, 2009 Publicado Javier Turégano Molina

Alta disponibilidad y balanceo de carga en Linux

http://informatica.gonzalonazareno.org/proyectos/2011-12/jlsc.pdf

Junio 22, 2012 Creador Juan Luis Sánchez Crespo

Cluster web en alta disponibilidad con lvs

http://serverfault.com/questions/173391/lvs-vs-haproxy-which-should-i-choose

Agosto 23, 2010

LVS vs HAProxy, which should i choose?

https://docs.google.com/presentation/d/1nrU0Q6SUOx12mHjOm76g6qJt9dXpiPTJEwLcB78GZK8/edit#slide=id.p

Noviembre 2012

Taller Balanceadores: ldirectord vs haproxy

http://es.scribd.com/doc/57937293/67/Tipos-de-balanceo-de-carga

Mayo 13, 2011

Balanceo de carga en Centos

http://bitsandbps.net/sysadmin/highavailability/load-balancing-using-keepalived-and-haproxy-on-centos-6-4-part-i/

Julio 9, 2013

Load Balancing using keepalive and HAProxy on CentOS 6.4

http://tecnocacharrero.blogspot.mx/2009/01/soluciones-de-balanceo-de-carga.html

Enero 1, 2009

Soluciones de Balanceo de Carga

https://www.strsistemas.com/blog/taller-II-balanceadores-ldirectord-vs-haproxy

Noviembre 2, 2012

Taller II – Balanceadores: ldirectord vs haproxy

http://ibiblio.org/pub/Linux/docs/LuCaS/Manuales-LuCAS/doc-curso-salamanca-clustering/html/ch03.html

Noviembre 2002

Capitulo 3. Balanceo de Carga

http://www.pymesyautonomos.com/tecnologia/software-de-redundancia-para-servidores-linux

Noviembre 30, 2010

Software de redundancia para servidores Linux

http://www.cyclopaedia.es/wiki/Linux-Virtual-Server

Linux Virtual Server

Octubre 23, 2013

http://www.loadbalancer.org/load_balancing_methods.php

Load Balancer

https://www.netlinux.cl/soluciones-it/alta-disponibilidad

Alta disponibilidad

https://es.wikipedia.org/wiki/Cl%C3%BAster_%28inform%C3%A1tica%29#Clasificaci.C3.B3n_de_los_cl.C3.BAsteres

Julio 3, 2015

Cluster

http://www.definicion.org/mirror

Definición mirror

http://www.alegsa.com.ar/Dic/dns.php

Diciembre 5, 2010

Definición DNS

http://www.alegsa.com.ar/Dic/hot%20swap.php

Diciembre 5, 2010

Definición hot swap

http://www.tp-link.com/mx/products/details/?model=TL-R480T%2B

Router Balanceador de carga

https://supportforums.cisco.com/es/discussion/11736921

cisco rv082

http://kemptechnologies.com/es/server-load-balancing-appliances/loadmaster-2600/overview/

LM-2600 Balanceador de carga

http://www.desarrolloweb.com/articulos/513.php

Agosto 22, 2011

Que es un firewall

http://www.alcancelibre.org/staticpages/index.php/como-dhcp-lan

Abril 22, 2015

Protocolo DHCP

http://www.ujaen.es/sci/redes/proxy/

Noviembre 22, 2005

Servicio Proxy

http://modeloosiyprotocolos.galeon.com/

Modelo OSI

https://es.wikipedia.org/wiki/Transmission_Control_Protocol

Mayo 5, 2015

Transmission Control Protocol

http://www.alegsa.com.ar/Dic/back-end.php

Diciembre 5, 2010

Definición back-end

http://www.masadelante.com/faqs/que-significa-http

Definición http

http://www.ehowenespanol.com/definicion-cookies-informatica-hechos_104985/

Definición de cookies

https://es.wikipedia.org/wiki/Localizador_de_recursos_uniforme

Julio 18, 2015

Localizador de recursos uniforme

http://www.alegsa.com.ar/Dic/servicio%20web.php

Diciembre 5, 2010

Definición servicio web

https://es.wikipedia.org/wiki/Trama_de_red

Marzo 13, 2015

Definición trama de red

http://docente.ucol.mx/al980347/public_html/capas.htm

Capas del modelo OSI

http://ccia.ei.uvigo.es/docencia/SCS/0910/transparencias/Tema4.pdf

Octubre 2008

Servicios web

https://es.wikipedia.org/wiki/Front-end_y_back-end

Junio 23, 2015

Front-end y back-end

http://www.alegsa.com.ar/Dic/https.php

Julio 5, 2009

Definición de https

https://es.wikipedia.org/wiki/Transport_Layer_Security

Julio 13, 2015

Transport layer Security

http://serverfault.com/questions/331079/haproxy-and-forwarding-client-ip-address-to-servers

Noviembre 15, 2011

Haproxy and forwarding client ip address to servers

http://www.maestrosdelweb.com/balance-de-carga-con-haproxy-configuracion/

Abril 27, 2011

Balance de carga con HAProxy

http://www.serverphorums.com/read.php?10,357873

Agosto 3, 2011

option fowardfor

http://www.genbetadev.com/seguridad-informatica/que-son-y-para-que-sirven-los-hash-funciones-de-resumen-y-firmas-digitales

Enero 15, 2013

Función hash

https://es.wikipedia.org/wiki/Cookie_%28inform%C3%A1tica%29

Junio 26, 2015

Cookie (informática)

http://blog.trungson.com/?p=250

Noviembre 11, 2009

Haproxy vs LVS (layer 7 vs layer 4 load balancing)

http://blog.aodbc.es/2013/01/02/soluciones-de-balanceo-de-carga-load-balancers/

Enero 2, 2013

Soluciones de balanceo de carga (Load Balancers)

https://unpocodejava.wordpress.com/2013/07/03/que-es-un-balanceador-de-carga-que-es-haproxy/

Julio 3, 2013

¿Qué es un balanceador de carga? ¿Qué es haproxy?