ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DISEÑO E IMPLEMENTACIÓN DE UNA SOLUCIÓN DE ALTA DISPONIBILIDAD USANDO CLÚSTER Y VIRTUALIZACIÓN DE SERVIDORES WEB Y DE BASES DE DATOS PARA LAS APLICACIONES DE LA FIEC"

INFORME DE PROYECTO DE GRADUACIÓN

Previo a la obtención del Título de:

INGENIERO EN TELEMÁTICA

Presentado por:

Katherine Estefanía Campos Bustos

Y

LICENCIADO EN REDES Y SISTEMAS OPERATIVOS

Presentado por:

José Luis Vera Chávez

Guayaquil - Ecuador

AÑO – 2015

AGRADECIMIENTO

Agradezco en primer lugar a Dios, por darme la fortaleza y sabiduría para culminar de manera exitosa este proyecto. A mi director, Ing Rayner Durango, por el apoyo brindado y por haber sido una guía durante el desarrollo del mismo. A mi familia, por su apoyo incondicional y por las palabras de fortaleza dadas. A mis profesores, amigos y compañeros de trabajo, por la colaboración brindada durante el avance del proyecto.

Katherine Estefanía Campos Bustos

Agradezco primero a Dios por iluminar mi camino. Agradezco también a mis padres e hijo, quienes en todo momento me dieron su apoyo incondicional. A los profesores, amigos y a todos los que hicieron esto posible.

José Luis Vera Chávez

DEDICATORIA

A Dios, quien siempre ha sido mi guía, mi fortaleza y mi ayudador. A mi familia por estar conmigo siempre, creyendo en mí y motivándome a ser mejor cada día. A mis profesores, amigos y compañeros de trabajo, por el apoyo brindado.

Katherine Estefanía Campos Bustos

A toda mi familia, principalmente a mi hijo, por todo el apoyo y estimulo, por creer en mí y nuca dejarme solo y a todas las personas que han confiado en mí.

José Luis Vera Chávez

TRIBUNAL DE SUSTENTACIÓN

MSc. Sara Ríos Orellana

SUBDECANA DE LA FIEC

Msig. Rayner Durango

DIRECTOR DEL PROYECTO DE GRADUACIÓN

PhD. Cristina Abad Robalino

MIEMBRO DEL TRIBUNAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Informe, nos corresponde exclusivamente; y el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL"

(Reglamento de graduación de la ESPOL)

Katherine Estefanía Campos Bustos

José Luis Vera Chávez

RESUMEN

El presente proyecto consistió en implementar una solución de alta disponibilidad usando clúster y virtualización de los servidores web y de bases de datos de la FIEC. Para ello, se realizó un estudio de las plataformas de virtualización, las soluciones de clúster y de las aplicaciones que se ejecutan en la FIEC. Las plataformas de virtualización que se probaron fueron el rol Hyper-V de Windows Server 2012 R2 y el hipervisor Xen en los modos de virtualización completa y paravirtualización, instalado sobre el S.O. Fedora, las cuales fueron sometidas a pruebas de estrés. Las soluciones de clúster que se probaron fueron el stack conformado por Pacemaker, CMAN y Corosync y el complemento de alta disponibilidad de Red Hat, a las cuales se les inyectó dos tipos de fallo para medir el tiempo de recuperación de los servicios.

Luego de hacer las pruebas respectivas, se concluyó que la plataforma de virtualización ideal a utilizar era el hipervisor Xen en el modo de paravirtualización, (3,82% y 37,16% de menor consumo de memoria RAM que Hyper-V y Xen en modo de virtualización completa en la prueba ab, respectivamente; 6,96% menor consumo de memoria RAM que Hyper-V en la prueba stress; 0,95% y 2,27 % menor consumo de CPU que Hyper-V en la prueba stress y ab, respectivamente). La solución de clúster seleccionada fue el stack conformado por Pacemaker, CMAN y Corosync, debido a que presentó menor tiempo de recuperación (24,27 y 22,57 segundos más rápido que el complemento de alta disponibilidad de Red Hat en el clúster de servidores de bases de datos y servidores web respectivamente). Finalmente, se instalaron algunas de las aplicaciones que se analizaron en este documento.

ÍNDICE GENERAL

AGRADECIMIENTO II
DEDICATORIA III
TRIBUNAL DE SUSTENTACIÓN IV
DECLARACIÓN EXPRESA V
RESUMEN VI
ÍNDICE GENERAL VII
ABREVIATURAS X
ÍNDICE DE FIGURAS XII
ÍNDICE DE TABLAS XIV
INTRODUCCIÓN XVI
CAPÍTULO 1 1
1. ANTECEDENTES Y JUSTIFICACION 1
1.1. Identificación del Problema 2
1.2. Justificación 2
1.3. Objetivos 3
1.3.1. Objetivo General 3
1.3.2. Objetivos Específicos 3
1.4. Solución propuesta 4
1.5. Metodología 4
1.6. Limitaciones 5
CAPÍTULO 2 7
2. MARCO TEÓRICO 7

2.1. Virtualización 8
2.2. Clúster 16
CAPÍTULO 3 21
3. ANÁLISIS DE LAS DIFERENTES PLATAFORMAS DE VIRTUALIZACIÓN Y SOLUCIONES DE CLÚSTER. 21
3.1. Análisis de las aplicaciones Web y Bases de Datos de la FIEC que requieren la solución de alta disponibilidad con virtualización. 22
3.2. Análisis de las diferentes plataformas de virtualización. 27
3.3. Análisis de las diferentes soluciones de clúster. 38
CAPÍTULO 4 44
4. DISEÑO, IMPLEMENTACIÓN Y PRUEBA DE LOS ESCENARIOS DE VIRTUALIZACIÓN Y DE IMPLEMENTACIÓN DE CLÚSTER 44
4.1. Diseño de los escenarios de virtualización 46
4.2. Implementación y prueba de las diferentes plataformas de virtualización. 50
4.3. Diseño de los escenarios de implementación de clúster. 53
4.4. Implementación y Prueba de las diferentes soluciones de clúster usando técnicas de inyección de fallos. 61
CAPÍTULO 5 67
5. ANÁLISIS DE LOS RESULTADOS E IMPLEMENTACIÓN DE LA SOLUCIÓN 67
5.1. Análisis de resultados de los diferentes escenarios de virtualización y soluciones de clúster 68
5.2. Selección de la plataforma de virtualización y clúster para la solución de alta disponibilidad. 77
5.3. Diseño de la Solución de Alta Disponibilidad. 80

5.4. Implementación de la Solución. 86
CONCLUSIONES Y RECOMENDACIONES 87
GLOSARIO 91
ANEXOS 97
Anexo A: Estudio preliminar 98
Anexo B: Figuras 99
Anexo C: Herramientas de carga de trabajo y monitoreo 104
Anexo D: Procesos de instalación y archivos de configuración final 106
Anexo E: Scripts personalizados 120
Anexo F: Resultados de las Pruebas de Estudio 129
BIBLIOGRAFÍA 139

ABREVIATURAS

CLVM	Clustered Logical Volume Manager (Administrador de volúmenes lógicos agrupados)
CMAN	Cluster Manager (Administrador de clúster)
CRM	Cluster Resource Manager (Administrador de recursos de clúster)
DLM	Distributed Lock Manager (Administrador de bloqueos distribuido)
DRBD	Distributed Replicated Block Device (Dispositivo de bloque replicado distribuido)
FC	Fibre Channel (Canal de fibra)
GFS	Global File System (Sistema global de archivo)
HTTP	Hypertext Transfer Protocol (Protocolo de transferencia de hipertexto)
HVMs	Hardware-assisted fully virtualized machines (Máquinas virtuales completamente de hardware asistido)
iSCSI	Internet SCSI
KVM	Kernel-based Virtual Machine (Máquina virtual basada en el núcleo)
LVM	Logical Volume Manager (Administrador de volúmenes lógicos)

NTPD	Network Time Protocol Daemon (Demonio del protocol de tiempo de red)
PVMs	Paravirtualized Virtual Machines (Máquinas virtuales paravirtualizadas)
RAID	Redundant Array of Independent Disks (Conjunto redundante de discos independientes)
RDP	Remote Desktop Protocol (Protocolo de escritorio remoto)
SAN	Storage Area network (Red de área de almacenamiento)
SCSI	Small Computer System Interface (Interfaz de sistema para computadoras pequeñas)
SELinux	Security-Enhanced Linux
STONITH	Shoot the other node in the head (Dispara a los demás nodos en la cabeza)
VDI	Virtualization Desktop Infrastructure (Infraestructura de virtualización de escritorio)
VMM	Virtual Machine Manager (Monitor de máquina virtual)
XML	Extensible Markup Language (Lenguaje de marcas extensible)

ÍNDICE DE FIGURAS

Figura 2.1. Arquitecturas de Hipervisor Tipo 1 (a) e Hipervisor Tipo 2 (b) [15] 16
Figura 2.2. Grupos de Recursos [21] 19
Figura 3.1. Arquitectura Xen [27] 29
Figura 3.2. Diagrama de la arquitectura de Citrix Xen Server [30] 30
Figura 3.3. Arquitectura KVM [7] 31
Figura 3.4. Diagrama de la arquitectura de Citrix Xen Server [33] 33
Figura 4.1. Computadora de Escritorio HP [42] 45
Figura 4.2. Conmutador HP [43] 46
Figura 4.3. Diseño de escenarios de virtualización a) Hipervisor Hyper-V b) Hipervisor Xen 48
Figura 4.4. Herramientas de carga de trabajo (a) stress (b) ab 51
Figura 4.5. Herramienta de monitoreo sar (a) para cpu y (b) para ram 52
Figura 4.6. Clúster virtual de dos nodos 54
Figura 5.1. Promedio de Uso de memoria RAM 70
Figura 5.2. Promedio de Uso de CPU 72
Figura 5.3. Tiempos de recuperación en el clúster de servidores de bases de datos 74
Figura 5.4. Tiempos de recuperación en el clúster de servidores web 75
Figura 5.5. Tiempo de Recuperación vs Tiempo de Monitoreo en Pacemaker para la técnica de inyección de fallo a nivel de servicios 76
Figura 5.6. Diagrama de la topología lógica de la solución de alta disponibilidad para los servidores de la FIEC. 81

Figura 5.7. Diagrama de la topología física del diseño de alta disponibilidad..... 84
Figura B-1. Gestor de máquinas virtuales con Virt-manager..... 99
Figura B-2. Creación de máquinas virtuales con Xen (a) Modo de Virtualización Completa y (b) Modo de Paravirtualización..... 99
Figura B-3. Ejecución del comando ab 100
Figura B-4. Pantalla de ingreso a la Interfaz de Administración del clúster creado con el complemento de Alta Disponibilidad de RedHat..... 101
Figura B-5. Resumen de clúster creados con el complemento de Alta Disponibilidad de RedHat 101
Figura B-6. Nodos del clúster creado con el complemento de Alta Disponibilidad de RedHat 102
Figura B-7. Dominio de conmutación del clúster creado con el complemento de Alta Disponibilidad de RedHat 102
Figura B-8. Grupo de Servicios del clúster creado con el complemento de Alta Disponibilidad de RedHat 103

ÍNDICE DE TABLAS

Tabla 1. Resumen de Plataformas de Virtualización 34
Tabla 2. Tecnologías de Virtualización a Probar 47
Tabla 3. Escenarios de Virtualización a probar 49
Tabla 4. Escenarios de soluciones de clúster a probar 55
Tabla 5. Direccionamiento de los Nodos de los clústeres 56
Tabla 6. Escenarios para las técnicas de inyección de fallo 60
Tabla 7. Soluciones de clúster 62
Tabla 8. Resultados del promedio de consumo de memoria RAM 69
Tabla 9. Resultados del promedio de consumo de CPU por el sistema 71
Tabla 10. Resultados del promedio de consumo de CPU por el usuario 72
Tabla 11. Tiempos de recuperación en el clúster de servidores de bases de datos 74
Tabla 12. Tiempos de recuperación en el clúster de servidores web 75
Tabla 13. Tiempo de recuperación vs tiempo de monitoreo en Pacemaker para la técnica de inyección de fallo a nivel de servicios 76
Tabla 14. Parámetros del comando stress 104
Tabla 15. Parámetros del comando stress 105
Tabla 16. Resultados del porcentaje de consumo de memoria RAM para la tecnología de virtualización Xen en el modo de paravirtualización. 129
Tabla 17. Resultados del porcentaje de consumo de CPU para la tecnología de virtualización Xen en el modo de paravirtualización. 129
Tabla 18. Resultados del porcentaje de consumo de memoria RAM para la tecnología de virtualización Xen en el modo de virtualización completa 130

Tabla 19. Resultados del porcentaje de consumo de CPU para la tecnología de virtualización Xen en el modo de virtualización completa..... 131
Tabla 20. Resultados del porcentaje de consumo de memoria RAM para la tecnología de virtualización Hyper-V en el modo de virtualización completa..... 132
Tabla 21. Resultados del porcentaje de consumo de CPU para la tecnología de virtualización Hyper-V en el modo de virtualización completa..... 132
Tabla 22. Resultados del tiempo de respuesta del stack Pacemaker, CMAN y Corosync al realizar la técnica de inyección de fallo a nivel de servicios 133
Tabla 23. Resultados del tiempo de respuesta del complemento de alta disponibilidad de Red Hat al realizar la técnica de inyección de fallo a nivel de servicios 134
Tabla 24. Resultados del tiempo de recuperación vs tiempo de monitoreo en Pacemaker para la técnica de inyección de fallo a nivel de servicios en el clúster de servidores de bases de datos 135
Tabla 25. Resultados del tiempo de recuperación vs tiempo de monitoreo en Pacemaker para la técnica de inyección de fallo a nivel de servicios en el clúster de servidores web 135
Tabla 26. Resultados del tiempo de respuesta del stack Pacemaker, CMAN y Corosync al realizar la técnica de inyección de fallo a nivel de nodos..... 136
Tabla 27. Resultados del tiempo de respuesta del stack Pacemaker, CMAN y Corosync al realizar la migración de recursos de forma manual 137
Tabla 28. Resultados del tiempo de respuesta del complemento de alta disponibilidad de RedHat al migrar los recursos de forma manual..... 137

INTRODUCCIÓN

Un clúster es una agrupación de computadoras unidas entre sí mediante una red de alta velocidad, de tal manera que se visualiza como un equipo de mayor capacidad. Hay diferentes tipos de clúster, entre los cuales están el clúster de alta disponibilidad y el clúster de alto rendimiento o de balanceo de carga. El clúster de alta disponibilidad, permite que los servicios que se encuentran ejecutando, permanezcan siempre activos sin ninguna interrupción, logrando que los clientes puedan acceder a los servicios en todo momento.

Además en la actualidad, las plataformas de virtualización de servidores han brindado la facilidad del uso de los distintos tipos de clúster, minimizando los costos tan elevados al momento de la implementación de la infraestructura física de esta tecnología de alta disponibilidad. Este proyecto consiste en usar ambas tecnologías, virtualización y clúster para implementar una solución de alta disponibilidad para los servidores web y de bases de datos de la FIEC.

CAPÍTULO 1

1. ANTECEDENTES Y JUSTIFICACION

Los servicios tecnológicos que se ejecutan en los servidores, se espera que nunca fallen y siempre estén funcionando. Por lo tanto, se necesita una solución que permita lograr este objetivo y evitar que los usuarios no puedan realizar sus actividades. La Facultad de Ingeniería en Electricidad y Computación cuenta con un grupo de servicios tecnológicos ejecutándose en sus servidores como son: CONTROLAC, SATT, ARA, entre otros. Estos servicios son accedidos en cualquier momento por los profesores y alumnos de la facultad, necesitando que siempre se encuentren disponibles. Lamentablemente la facultad no cuenta con una solución de alta disponibilidad que permita que en caso de que falle el servidor donde se encuentran tales aplicaciones, estas puedan migrarse a otro equipo de manera que haya continuidad operacional y los usuarios no se vean afectados por el impedimento de poder acceder a los servicios y trabajar sobre

los mismos. Además, los servicios tecnológicos de la FIEC, solamente usan un plan de respaldo para proteger la información de las bases de datos. Esto implica que la solución de alta disponibilidad satisface este problema.

1.1. Identificación del Problema

Debido al tiempo de funcionamiento que tiene el hardware sobre el cual se ejecutan las aplicaciones de la FIEC, en varias ocasiones se ha tenido que interrumpir los servicios que ésta presta, para realizar el mantenimiento preventivo y correctivo de los servidores. Al no contar con la solución de alta disponibilidad los usuarios se han visto afectados por el tiempo de interrupción de los servicios.

1.2. Justificación

A través de la implementación de una solución de alta disponibilidad usando las tecnologías de las plataformas de virtualización, el Departamento de Soporte Técnico de la Facultad de Ingeniería en Electricidad y Computación podrá contar con un plan de prevención y continuidad para los servicios tecnológicos hospedados en los servidores. La tecnología de la virtualización ayudará a colaborar con las implementaciones de alta disponibilidad, por la flexibilidad en el manejo de las máquinas virtuales. Superando el problema de la interrupción de los servicios que se encuentran utilizando en la facultad. Con la utilización de la virtualización de servidores se deberá implementar al menos “dos servidores host”

respetando la tecnología de clúster al manejar un nodo activo y pasivo. De esta forma se eleva la confiabilidad en la solución de virtualización y clúster.

1.3. Objetivos

Los Objetivos que alcanzaremos en este proyecto serán los que se muestran a continuación, describiendo en el objetivo general la idea completa de lo que se desea lograr y en los objetivos específicos las metas concretas que permitirán detallar mejor el objetivo general.

1.3.1. Objetivo General

Diseñar e implementar una solución de alta disponibilidad para los servidores web y de bases de datos de la Facultad de Ingeniería en Electricidad y Computación (FIEC), que brinde continuidad operacional.

1.3.2. Objetivos Específicos

- Analizar y diseñar la infraestructura de servidores virtualizados para la FIEC.
- Implementar y analizar entre las diferentes soluciones de clúster que se usan para las plataformas de virtualización.
- Realizar las pruebas de las plataformas de virtualización y de clúster para las aplicaciones de los servidores web y bases de datos.

- Diseñar e implementar la solución de alta disponibilidad para las aplicaciones de la FIEC, en base a las pruebas realizadas de las plataformas de virtualización y de clúster.

1.4. Solución propuesta

Implementar una solución de clúster de alta disponibilidad para los servidores web y de bases de datos que maneja la FIEC, ya que se espera que los servicios tecnológicos que se encuentran alojados y se ejecutan en tales servidores siempre estén funcionando y no fallen o al menos que el tiempo de inactividad sea muy bajo. Además que esta solución de clúster se realicé sobre máquinas virtuales alojadas en los servidores físicos, de tal forma que halla un mayor ahorro del consumo de los recursos del hardware.

1.5. Metodología

El procedimiento para el desarrollo del proyecto es:

1. Análisis de las aplicaciones y/o servicios web que maneja la FIEC para determinar cuáles son aquellas que requieren una mayor disponibilidad, y cuáles son los requerimientos de las mismas al momento de implementar la solución de alta disponibilidad.
2. Investigar y analizar las diferentes soluciones de clúster y virtualización que existen y se acoplen al hardware proporcionado por el DST, que consiste en dos computadores de escritorio de última tecnología, y

permitan brindar una solución de alta disponibilidad a las aplicaciones de la FIEC.

3. Realizar un estudio comparativo con respecto al rendimiento y alta disponibilidad de las soluciones de clúster y virtualización que se investigaron, mediante técnicas de inyección de fallos, para así determinar la solución de alta disponibilidad adecuada.
4. Implementar la mejor solución de alta disponibilidad según el estudio realizado, con los requisitos que necesiten las aplicaciones que maneja la FIEC.

1.6. Limitaciones

A lo largo del proyecto, se presentaron ciertas limitaciones, las cuales influyeron en el desarrollo del mismo. Una de ellas fue el hardware que se utilizó para las pruebas, ya que a pesar de ser potente y de última tecnología, carecía de ciertas características que el hardware de un servidor suele tener. Además ciertos componentes de hardware como la tarjeta de red no era compatible con una de las soluciones de virtualización que se deseaba probar la cual es VMWare, ya que la versión del controlador Intel no estaba soportado por esta plataforma. Otra de las limitantes fue que los equipos de cómputo sobre los cuales se instalaron los nodos del clúster, tenía una interfaz de red integrada y solo poseía un slot para agregar una tarjeta de red adicional, lo cual permitió tener solo dos interfaces de red e impidió con una tercera tarjeta de red, crear una red redundante que fortaleciera la implementación del clúster en la solución de alta

disponibilidad. Finalmente el número de IPs públicas con el que se contó estuvo limitado, de manera que no fue posible configurar una IP pública para todos los nodos del clúster, sino solo para los nodos del clúster de Servidores Web.

CAPÍTULO 2

2. MARCO TEÓRICO

En la actualidad la Facultad de Ingeniería en Electricidad y Computación requiere que los servicios web que presta estén disponibles los 365 días del año, debido a que a través de ellos se realizan trámites de vital importancia para la facultad e institución. Es por ello que se necesita de una solución de alta disponibilidad que permita eliminar o minimizar los tiempos de inactividad de tales servicios que puedan suscitarse debido a algún fallo de hardware, sobrecarga de tráfico o a algún mantenimiento programado. Para lograr este objetivo se hará uso de dos tecnologías muy importantes en la actualidad, las cuales son virtualización y clúster.

La virtualización es una tecnología que permite crear una adaptación virtual de un recurso tecnológico mediante el uso de software. Fue desarrollada por la compañía IBM® en los años 60, y tuvo una gran acogida durante esa década y la

siguiente. Sin embargo en la década de los 80 tuvo un declive debido a la producción de hardware más barato y sistemas operativos multiusuario. Fue a partir de los 90, que se volvió a emplear esta tecnología debido a las necesidades para las cuales era requerida entre las cuales están un mejor uso de los recursos físicos, la capacidad de hacer pruebas en un entorno virtual sin afectar componentes físicos, el ahorro de costos, entre otros [1][2].

Por otro lado, clúster es la agrupación de varios equipos de computación que mediante software especializado funciona como si se tratase de un solo equipo con mayor capacidad. Esta tecnología puede ser utilizada tanto para brindar alta disponibilidad, balanceo de carga o mejorar el rendimiento de ciertos procesos. A lo largo de este capítulo se detallaran los conceptos relevantes de estas tecnologías, de manera que el lector pueda comprender el desarrollo e implementación que involucra nuestro proyecto para proveer a la facultad de la solución de alta disponibilidad que necesita.

2.1. Virtualización

La virtualización es una tecnología que permite abstraer los recursos de un computador, mediante el uso de una capa lógica que es la que se encarga de gestionar y entregar los recursos de hardware a los clientes que están siendo virtualizados. Esta tecnología oculta detalles técnicos a través de la encapsulación. Un ejemplo de virtualización es la simulación de equipos de cómputo físicos a través del uso de máquinas virtuales, permitiendo que múltiples instancias de sistemas operativos corran de manera concurrente en un solo computador, abstrayendo la mayor cantidad de aplicaciones

posibles y protegiéndolas al colocarlas en máquinas virtuales diferentes [3] [2] [4].

Además, la virtualización produce un gran ahorro de costos en los sectores de computación, especialmente en el área de tecnologías de información, debido a que en una granja de servidores es mucho más económico y fácil dar mantenimiento a unas pocas máquinas físicas que contienen máquinas virtuales [5]. Otra de las ventajas que da la virtualización es que permite hacer un mejor uso de los recursos principales de un computador como son CPU, red, almacenamiento y memoria, además de la optimización de la energía, ahorro de espacio físico, mejor disponibilidad, mayor seguridad y la administración simplificada y unificada (centralización). Sin embargo, si llega a fallar el hardware que se utiliza para virtualizar, pueden fallar varios servicios hospedados en las máquinas virtuales al mismo tiempo. Otra de las desventajas de la virtualización es que el diseño previo a la implementación es más laborioso, además de que puede haber mucha congestión o uso excesivo de los recursos de red y la mayor complejidad de las tareas de administración en general [6].

Existen varios tipos de virtualización, entre los cuales se puede mencionar la **Virtualización de recursos individuales**, la cual involucra la simulación de recursos como volúmenes de almacenamiento, espacios de nombres y recursos de red. Como ejemplo de este tipo de virtualización tenemos a los Discos *RAID* y *gestores de volúmenes* (LVM), virtualización de almacenamiento como *SAN*, redes privadas virtuales, sistema

multiprocesador y multinúcleo, clúster, computación grid y computación en la nube [2].

Otro tipo de virtualización es la **Virtualización a nivel de sistema operativo,** también llamada virtualización basada en contenedores, la cual extiende el concepto de *chroot* para conseguir aislar los diferentes entornos de ejecución que las aplicaciones ven como máquinas virtuales [7]. Los huéspedes aislados se denominan contenedores [8]. En este tipo de virtualización solo se ejecuta el núcleo del anfitrión, el cual ofrece esa funcionalidad del sistema operativo a cada uno de los huéspedes [9]. Es decir, todos los sistemas invitados se ejecutan sobre un mismo núcleo, con el mismo sistema operativo y comparten recursos con otros contenedores (máquinas virtuales), aunque el sistema le hace creer a las aplicaciones que están ejecutándose en sistemas independientes. Este comportamiento convierte a este tipo de virtualización en una solución económica, puesto que no necesita apoyo de hardware ni la supervisión a bajo nivel, la cual es una tarea que ejecuta el *hipervisor*. Sin embargo, si llegase a fallar el *núcleo* todos los entornos virtuales fallarían. A pesar de que no es necesario emular el hardware a bajo nivel, la virtualización a nivel de sistema operativo incluye apoyo para tener dispositivos virtuales como discos o tarjetas de red en cada contenedor [7]. Esta técnica solo se puede aplicar a sistemas de distribución libre, debido a que solo en ellas se puede realizar modificaciones en el núcleo del sistema operativo anfitrión. Los proveedores de alojamiento se benefician de esta técnica, debido a que necesitan una manera eficiente y segura para ofrecer sistemas operativos

iguales a sus clientes [8]. Sin embargo aquellos entornos donde es necesario tener varios sistemas operativos diferentes es preferible trabajar con *hipervisores.*

La **Virtualización de aplicaciones**, es otro tipo de virtualización y consiste en aislar la componente lógica de la aplicación del componente sistema operativo. El objetivo es conseguir que las aplicaciones puedan funcionar con independencia de las características concretas del entorno en que se ejecutan [10].

También tenemos a la **Virtualización de Escritorio** como un tipo de virtualización, la cual consiste en un proceso de separación entre el escritorio que engloba los datos y programas que utilizan los usuarios de la máquina física, lo cual se debe a que el escritorio virtualizado se almacena de manera remota en un servidor central [1]. Los escritorios virtualizados pueden alojarse además en servidores dentro de un centro de datos; lo que se conoce como una infraestructura de escritorio virtual *(VDI)* [11]. En este tipo de virtualización el cliente no necesita tener equipos de última tecnología, porque todas las aplicaciones del escritorio son ejecutadas centralmente en un servidor remoto, lo cual permite un ahorro de costos que se reducen aún más si para acceder a los escritorios virtuales se utiliza clientes ligeros, que son más económicos que las computadoras tradicionales. La comunicación entre los dispositivos de acceso y los servidores de escritorio, se realizará mediante protocolos especiales, en función del tipo de escritorio que se utiliza. Por ejemplo, si se desea tener

escritorios Microsoft Windows se utilizará el protocolo RDP y si se quiere escritorios Linux el protocolo a usar será X Window [12].

El último tipo de virtualización que mencionaremos será la **Virtualización de plataforma,** la cual se lleva a cabo en una plataforma de hardware mediante un software de virtualización anfitrión, el cual es un programa de control que simula una o varias máquinas virtuales, sobre la cual se instala el sistema operativo invitado de la misma forma como si lo estuviese haciendo en una máquina real. En lugar de comprar varios servidores para que realicen funciones específicas, la virtualización de plataforma permite correr varios servidores virtuales en un solo equipo físico, lo cual conlleva a que las cargas de trabajo se consoliden en un número más reducido de servidores y estos sean plenamente utilizados [11].

A su vez la virtualización de plataforma se divide en Emulación, Virtualización Completa y Paravirtualización. La ***Emulación*** es una técnica de virtualización que simula una arquitectura completa de hardware a través de un software de emulación especializado, permitiendo que sistemas operativos que fueron escritos para otra plataforma o arquitectura diferente al del equipo anfitrión puedan ser ejecutados en el mismo, dado que traduce cada instrucción de la máquina virtual a una instrucción válida para la arquitectura de la máquina anfitrión [13]. El programa de emulación tiene el control total de los sistemas emulados, incluyendo la ejecución de instrucciones a nivel de CPU [7]. Para escenarios en los que el tiempo de respuesta no es un factor importante, la emulación podría ser la mejor

opción debido a que el tiempo de respuesta para resolver una petición de algún recurso es alto, presentando penalizaciones en el rendimiento. Como ejemplos de emulación tenemos a Bochs, MAME y QEMU.

La **Virtualización completa**, es aquella que se encarga de emular el hardware en toda su extensión de forma tal que el sistema operativo invitado puede correr en la plataforma de virtualización sin ser modificado [14]. El software de virtualización que utiliza es conocido como Monitor de Máquina Virtual (VMM) o Hipervisor y se encarga de emular un sistema completo y analizar dinámicamente el código que quiere ejecutar el sistema invitado (éste concepto será ampliado al finalizar éste capítulo de manera que el lector pueda comprender a fondo la función del hipervisor). Se diferencia de la emulación, debido a que no traduce instrucciones de CPU sino que las ejecuta directamente, lo que impide que un sistema operativo diseñado para una arquitectura diferente, pueda ejecutarse sobre el sistema anfitrión [13]. La virtualización completa a su vez se divide en Virtualización nativa sin apoyo de Hardware y Virtualización nativa con apoyo de hardware, cuya diferencia radica en que la Virtualización nativa con apoyo de hardware aprovecha tecnologías incorporadas a las nuevas generaciones de microprocesadores como las de Intel® y AMD® para poder ejecutar el código del sistema operativo sin modificarlo; estos sistemas ejecutan el Hipervisor o VMM con el máximo nivel de acceso a la CPU (*anillo* -1 en procesadores AMD® e Intel®) y los sistemas invitados se ejecutan a un nivel inferior (Anillo 0 en procesadores AMD® o Intel®, que era el máximo nivel de ejecución cuando los procesadores no incorporaban

apoyo para la virtualización). Un ejemplo de este tipo de virtualización es KVM. Como ejemplos de virtualización sin apoyo de hardware se puede mencionar a VirtualBox y VMWare [7].

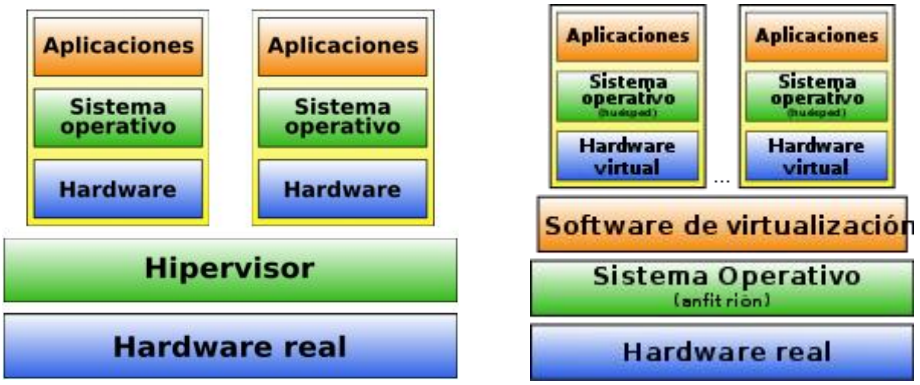
Por otra parte, la **Paravirtualización** consiste en modificar el sistema operativo invitado para incluir las instrucciones relacionadas con la virtualización. La desventaja de este sistema es que no siempre es posible modificar el sistema operativo invitado, como es el caso de Microsoft Windows. Al modificar el sistema operativo invitado el hipervisor ya no captura las instrucciones problemáticas, sino que el invitado llama directamente al hipervisor cuando es necesario [7]. Debido a que no requiere de una emulación completa de los dispositivos o recompilación dinámica para atrapar instrucciones privilegiadas, la Paravirtualización a menudo funciona a una velocidad cercana de la nativa [14].

Luego de haber conocido los diferentes tipos de virtualización que existen se puede llegar a la conclusión que la Virtualización de Plataforma, de manera más específica la Virtualización de Plataforma Completa y la Paravirtualización, jugarán un papel muy importante al momento de implementar la solución de alta disponibilidad. Un concepto que ambos tipos de virtualización de plataforma comparten es el de *hipervisor,* también conocido como monitor de máquinas virtuales (VMM), el cual es considerado como la capa lógica entre los recursos físicos de un ordenador y los del cliente, quienes realizan peticiones para usar tales recursos. Permite ejecutar varias máquinas virtuales distintas, con sistemas

operativos diferentes, sobre una máquina real, ya que puede correr procesos que requieran de sistemas operativos diferentes, lo cual permite aprovechar las cualidades propias de cada sistema operativo, sin tener que cambiar de máquina.

Existen dos tipos de hipervisores. El primero de ellos conocido como **Hipervisor Tipo 1**, también denominado como Hipervisor nativo o bare-metal, es el software que se instala directamente sobre el hardware y cumple las funciones tanto de sistema operativo como de virtualización [15]. Su rendimiento es mayor al del tipo 2, debido a que actúa directamente sobre el equipo. La Figura 2.1 (a) muestra la arquitectura de este tipo de hipervisor. Los hipervisores más conocidos del tipo 1 son: Hyper-V, Citrix XenServer, Oracle VM y VMware Sphere.

Por otro lado el **Hipervisor de Tipo 2**, o también conocido como hosted, es el software que se ejecuta sobre un sistema operativo existente. Es la manera de virtualizar menos eficiente, debido a que los recursos del equipo virtualizado los controla el sistema anfitrión y existe una sobrecarga al pasar por una capa adicional, como se observa en la figura 2.1. (b), además de no ser tan seguro puesto que si el sistema operativo anfitrión es afectado, el sistema virtualizado también lo será. Sin embargo, se puede seguir utilizando otras aplicaciones en el equipo físico. Ejemplos de este tipo de Hipervisor son VMware Workstation, Oracle VirtualBox, VirtualBox OSE, QEMU, Virtual PC, Virtual Server.



(a) (b)

Figura 2.1. Arquitecturas de Hipervisor Tipo 1 (a) e Hipervisor Tipo 2 (b) [15]

2.2. Clúster

Clúster es la unión de múltiples computadoras independientes dentro de un sistema, a través del uso de software y redes de alta velocidad, que se comportan como si fuesen una única computadora. Por definición un clúster debe tener al menos dos computadoras, denominadas nodos. Los nodos del clúster se comunican usando una interfaz de paso de mensajes [16]. Son utilizados especialmente para mejorar el rendimiento y/o la disponibilidad que un único sistema no puede alcanzar. Además de ser una alternativa económica y que puede funcionar de manera similar a costosos sistemas de alta potencia y disponibilidad [17]. Para que el clúster funcione, no basta con conectar los equipos entre sí, sino que requiere de un sistema que se encargue de administrarlo. De manera general el clúster necesita de varios componentes para funcionar, entre los cuales podemos mencionar a los *nodos, sistema operativo, middleware, servicios y*

aplicaciones, conexiones de red, protocolos de comunicación y sistemas de almacenamiento.

Además, basándose en las necesidades del usuario, los tipos de clúster que existen según su funcionalidad son clúster de almacenamiento, clúster de alta disponibilidad, clúster de balanceo de carga y clúster de alto rendimiento. El **Clúster de Almacenamiento** proporciona una imagen de sistema de archivos a lo largo de los servidores en el clúster, permitiendo que los servidores lean y escriban de forma simultánea a un sistema de archivos compartido. La ventaja de usar un sistema de archivos en todo el clúster, es que se elimina la necesidad de copias excesivas de los datos de la aplicación y se simplifica la creación de copias de seguridad y recuperación contra desastres [18].

Por otra parte el **Clúster de Alta Disponibilidad** es aquel que proporciona continua disponibilidad de los servicios, mediante el proceso de recuperación de fallos, al trasladar el servicio desde el nodo erróneo del clúster a otro nodo completamente funcional. Este tipo de clúster debe mantener la integridad de los datos cuando un nodo recibe el control del servicio desde otro nodo. El proceso de traslado de servicios es completamente transparente al cliente. También se los conoce como clúster con recuperación de fallas [18]. Entre las ventajas que se tiene al usar este tipo de clúster se puede mencionar el aumento de la disponibilidad, la mejora del rendimiento, la escalabilidad, la tolerancia a

fallos, la recuperación ante fallos en tiempo aceptable, la reducción de costos, la consolidación de servidores y del almacenamiento.

Otro tipo de clúster es el de **Balanceo de Carga**, el cual responde a peticiones de servicios de red desde diferentes nodos para balancear las peticiones a lo largo de los nodos del clúster. Si un nodo del clúster falla, el software de balance de carga detecta la falla y asigna las peticiones a otros nodos en el clúster. Asimismo el proceso de balanceo de carga es transparente al cliente [18].

Por último tenemos el **Clúster de Alto Rendimiento,** el cual utiliza los nodos para ejecutar cálculos simultáneos. Permite que las aplicaciones trabajen de forma paralela, mejorando así el rendimiento de éstas. Se los conoce como clúster computacionales o computación de red [18]. Podría darse el caso que de acuerdo a la necesidad del cliente se combinen los tipos de clúster descritos anteriormente.

Debido a sus características, el clúster de alta disponibilidad será el que nos ayudará a cumplir el objetivo de implementar la solución de alta disponibilidad que requiere la FIEC. Es por esta razón que detallaremos el funcionamiento del mismo. La primera función del clúster de alta disponibilidad es comunicar entre sí a los nodos, monitorizando continuamente su estado y detectando fallos. Para poder realizar aquello es habitual usar un canal específico para la comunicación, la cual puede ser una red IP independiente o una conexión serie, de manera que la misma

no se vea afectada por problemas de seguridad o rendimiento. Además utiliza una técnica llamada ***Heartbeat,*** como se muestra en la figura 2.2.

La segunda función es administrar los servicios ofrecidos por el clúster, teniendo la capacidad de migrar dichos servicios entre diferentes servidores como respuesta a un fallo [19], según se muestra en la Figura 2.2. Para llevar a cabo esta función es necesario realizar seguimientos a nivel de recursos o servicios y detectar el fallo de los mismos, donde el Administrador será quien configure la periodicidad del monitoreo y las acciones que se deberán llevar a cabo. Este proceso se conoce como **Monitoreo de Recursos.** Cabe recalcar que el término *recurso* en este contexto se aplica a cualquier componente físico o lógico, administrable en un clúster y que solo se puede alojar en un nodo a la vez [20], el cual o los cuales son provistos a los servicios para que realicen una tarea específica. Como ejemplos de recursos tenemos scripts de arranque del servicio, un sistema de ficheros, una dirección IP, etc.

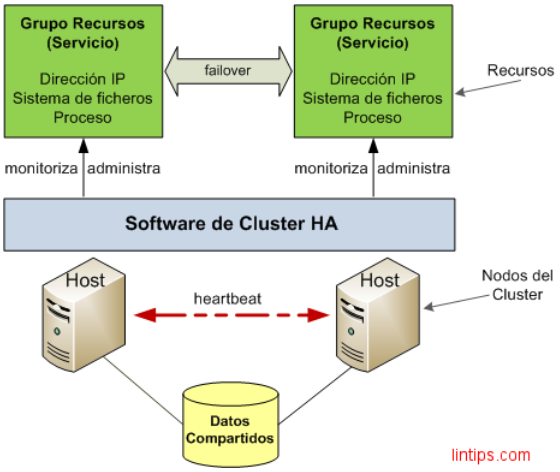


Figura 2.2. Grupos de Recursos [21]

El administrador también podrá definir la preferencia de nodos, la cual ayuda a distribuir los servicios entre los diferentes servidores de acuerdo al hardware y según interese para obtener el resultado ideal del clúster. En el caso de que suceda un fallo, la primera medida a tomar será la de reiniciar los recursos en el mismo nodo. Sin embargo se debe tomar en cuenta que dentro de un clúster, un servicio puede requerir más de un recurso, los cuales se agrupan y son dependientes entre sí, por lo que se requiere que al momento de arrancar o detener un servicio, los recursos se activen en el orden adecuado. En el caso que no sea posible reiniciar los recursos en el mismo nodo, se procederá a migrarlos. Para ello, el software de clúster abstrae e independiza a los servicios de un nodo concreto, permitiendo que se desplacen entre diferentes servidores de manera transparente para la aplicación o los usuarios. El tiempo de inactividad por el posible fallo es mínimo y el clúster continuará proporcionando el correspondiente servicio [16].

En el caso de que un nodo del clúster esté funcionando de manera incorrecta, el clúster hará uso del *Fencing*. Durante el funcionamiento del clúster también se puede presentar el escenario *Split-Brain*. Una de las posibles causas para que se genere este escenario es la pérdida de conexión entre los nodos, lo que provoca que el clúster se divida, y cada una de las particiones que se forma, crea que es única e intente activar los servicios. Para remediar aquello se aplica un mecanismo denominado *Quorum*.

CAPÍTULO 3

3. ANÁLISIS DE LAS DIFERENTES PLATAFORMAS DE VIRTUALIZACIÓN Y SOLUCIONES DE CLÚSTER.

Como se mencionó en la introducción del capítulo anterior, es importante que los servicios que presta la Facultad de Ingeniería en Electricidad y Computación estén siempre disponibles. Es por ello, que a lo largo de esta sección se hará un análisis de las aplicaciones que proveen de tales servicios a la facultad. Asimismo, luego de haber realizado un estudio de los conceptos fundamentales de las tecnologías que serán implementadas en nuestro proyecto, durante este apartado se analizarán las tecnologías más relevantes que existen en la actualidad, para así poder llegar a la conclusión de cuáles serán aquellas que nos ayudarán a cumplir el objetivo de implementar una solución de alta disponibilidad.

3.1. Análisis de las aplicaciones Web y Bases de Datos de la FIEC que requieren la solución de alta disponibilidad con virtualización.

Es importante tomar en cuenta que día a día aumenta la necesidad de la automatización de procesos a través del uso de aplicaciones web, lo cual obliga al Departamento de Soporte Técnico garantizar la alta disponibilidad tanto de las aplicaciones actuales como de las que se implementarán en un futuro. Actualmente, las aplicaciones con las que cuenta la FIEC, trabajan en su mayoría sobre un servidor Apache y el restante trabaja sobre un servidor Apache Tomcat con su respectiva base de datos, las cuales en algunos casos están en MySQL y otras en PostgreSQL. A continuación se realizará un breve análisis de cada una de las aplicaciones Web de la FIEC, el cual permitirá comprender más a fondo la necesidad de implementar la solución de alta disponibilidad, debido a que una interrupción del servicio que prestan estas aplicaciones podría ocasionar el atraso de muchos trámites, causando molestias tanto a estudiantes, como a docentes y al personal administrativo.

Controlac (Control Académico). Este sistema sirve para que los docentes que dictan materias dentro de las instalaciones de FIEC registren las clases. La finalidad de este sistema es proveer un reporte de las clases dictadas a los coordinadores de carreras, a su vez al final del semestre los docentes pueden generar un reporte con todas las clases que han ingresado al sistema, dicho reporte se genera en un excel, el mismo que

pueden subir en el académico. Este sistema funciona en un servidor web con el servicio de ruby corriendo, está escrito en lenguaje Ruby y utiliza lo que se conoce como gemas para su correcto funcionamiento, utiliza una base de datos mysql.

Nemesis. Este sistema fue hecho para que los docentes pudieran llevar control de los proyectos que elaboraban los usuarios, el sistema permite la creación de un proyecto y la inclusión de integrantes (estudiantes); los integrantes pueden interactuar entre ellos, crear actividades en el calendario y todas las actividades que realicen podrán ser monitoreadas por el docente a cargo del proyecto. Este sistema funciona en un servidor web apache, está implementado en php y funciona con una base de datos mysql.

CRM. Este sistema no es una implementación propia del DST-FIEC, sino más bien una herramienta adaptada a las necesidades de la FIEC, el sistema original se conoce como CRM vtiger [22]. El módulo de este sistema que se utiliza en la FIEC es el de “Tickets Support” el cual permite a un usuario del sistema crear un ticket en el cuál describe el problema asociado a una de las categorías dentro del sistema, asigna el ticket a alguno de los usuarios dentro del staff de soporte técnico y dicho usuario será el encargado de reasignar el ticket o de atenderlo en el caso que esté dentro de sus funciones y conocimientos para resolver el problema. El sistema provee una interfaz para atender los tickets y otra interfaz que está solo dedicada a los usuarios que crean los tickets, la segunda interfaz se

conoce como “portal”, aquí solo se pueden crear y consultar los tickets ya creados, no se los puede atender ni modificar. Funciona en un servidor web apache y está implementado en php utiliza una base de datos mysql.

SCDoc. Este sistema también conocido como “workflow” sirve para gestionar documentos. Es decir cualquier documento que se necesitaba pasar a secretaría, decanato o subdecanato debía subirse a este sistema. Con la utilización del sistema Quipux, este sistema ha quedado casi sin uso. Este sistema funciona en un servidor web apache, está implementado en php y utiliza una base de datos mysql.

Certifiec. Sistema que sirve para la emisión de certificados, los usuarios que pueden acceder al sistema son aquellos estudiantes cuyo estado sea activo en los web services ESPOL. La finalidad de este sistema es que el estudiante pueda solicitar el certificado online y después de un determinado número de días reciba un correo notificándole que puede acercarse a secretaria a retirar su certificado. Funciona en un servidor web apache, está implementado en php, utiliza una base de datos mysql.

Creación de cuentas. Sistema implementado para la automatización de creación de cuentas de usuarios de la FIEC, el sistema crea un usuario en la base de datos de la FIEC y le da acceso a todos los beneficios de tener una cuenta google (correo, drive, etc). Funciona con un servidor web apache y está implementado en php.

Recuperación de contraseña. Sistema modificado del antiguo sistema de recuperación de contraseñas, sirve tanto para cambiar la contraseña de la cuenta FIEC como para recuperar la contraseña en caso de haberla olvidado (recuperar funciona solo para estudiantes). Funciona en un servidor web apache y está implementado en php.

ARAFIEC (Aplicación para la Recepción de Artículo). Sistema que sirve para que los estudiantes que hayan terminado su tesis y hayan sustentado, puedan subir el resumen de la tesis en formato de achivos pdf al sistema. El sistema evita que los estudiantes se acerquen a secretaria a entregar el resumen. El proceso como tal es el siguiente: el estudiante sube su resumen, una vez llenado los datos, el proceso comienza enviándose a la interfaz del docente el resumen para que el docente lo revise y proceda a aceptarlo, una vez aceptado el resumen pasa a la interfaz de la secretaria la cual tomará todos los datos necesarios para crear el informe que posteriormente enviará a secretaría técnica académica para el desbloqueo de la deuda de no valor en el académico. Está implementado usando una herramienta llamada WebRatio que utiliza java como lenguaje.

Sistema de Reservar Salas. Es un sistema, a través del cual se puede reservar los diferentes laboratorios que existen en la Facultad para el dictado de clases. El sistema original se conoce como MRBS [23]. Está implementado usando un lenguaje php y trabaja sobre un servidor Apache y usa una base de datos MySQL.

SATT (Sistema de aprobación de temas y temarios). Es un sistema web que permite ingresar una solicitud para la aprobación de un tema y/o temario para graduarse en una de las modalidades que ofrece la FIEC. El proceso que se desarrolla una vez generada la solicitud e ingresada la información necesaria para un proyecto de graduación o tesis, involucra varios perfiles: estudiante, secretaria, docente (que a su vez puede ser un evaluador o el director de la tesis como tal), y subdecano(a) que es la persona encargada de garantizar que el proceso de graduación se realice de manera correcta. El estudiante es quien inicia el proceso en el SATT, ingresando su información personal y los detalles de su tesis, también escoge a un director de tesis, el cual será el encargado de dar retroalimentación al estudiante respecto al trabajo realizado. El estudiante durante esta parte puede realizar modificaciones a la información y documento ingresado inicialmente.

Cuando el director aprueba la solicitud, se asignan evaluadores al trabajo realizado por el o los estudiantes quienes emiten un informe de la solicitud una vez revisada la información del proyecto o tesis, posteriormente se asignan a los miembros del tribunal. El subdecano a cargo es quien revisará los informes de los evaluadores. Posteriormente los miembros del consejo directivo votan por la aprobación o no de la solicitud, si no es favorable se emite un informe de los cambios que el o los estudiantes deben realizar, para ser revisados y tratados nuevamente en una reunión del consejo directivo, finalmente si el consejo aprueba la solicitud, el estudiante puede proceder con el proceso de sustentación de su tesis para finalmente

graduarse y obtener el título de pregrado. El sistema fue implementado con la herramienta WebRatio que a su vez utiliza Java como lenguaje y una base de datos mysql.

3.2. Análisis de las diferentes plataformas de virtualización.

Como se indicó en el capítulo anterior existen varios tipos de virtualización. En esta sección se procederá a analizar aquellas soluciones que tengan un mejor rendimiento, desarrollo, soporte, facilidad de adquisición al usuario, entre otros factores importantes a tomar en cuenta y que se ajusten a las necesidades de la Facultad permitiendo la virtualización de los sistemas que fueron analizados previamente. A continuación se hará un breve estudio de cada una de estas tecnologías.

Denali. Es una plataforma de paravirtualización, que puede definirse como un monitor de máquina virtual IA-32, que permite que servicios que no son de confianza se ejecuten en dominios aislados. El modelo original de Denali fue soportar máquinas corriendo servicios ligeros de internet. Una revisión posterior incluye soporte para ejecutar sistemas operativos con numerosas funciones [24] [1].

Xen. Es un Monitor de Máquina Virtual de código abierto desarrollado por la Universidad de Cambridge, que permite ejecutar varias instancias de un sistema operativo o a su vez varios sistemas operativos diferentes en paralelo en un mismo equipo físico sin sacrificar tanto el rendimiento como la funcionalidad [25]. Para ello provee de una abstracción idealizada de

máquina virtual que permite que sistemas operativos como Linux, BSD y Windows puedan ser alojados en tales máquinas con el mínimo esfuerzo. Soporta la paravirtualización y la virtualización completa de hardware asistido y permite que tanto las PVMs como las HVMs puedan correr al mismo tiempo. El hypervisor Xen debe iniciar antes que cualquier núcleo de Linux debido a que es el encargado de controlar el hardware y distribuir su uso entre las diversas máquinas virtuales.

Xen denomina a las máquinas virtuales dominios, los cuales deben correr en el núcleo de Xen en vez del núcleo por defecto y pueden ser de dos tipos: Dom0 y DomU. Dom0 o también conocido como dominio 0, es el dominio administrativo privilegiado que Xen lo inicia al arrancar, y del cual solo uno puede ejecutarse, como se muestra en la Figura 3.1. [26]. Como su nombre lo indica, ejecuta las herramientas de administración de Xen y tiene privilegios especiales como habilitar el acceso al hardware de manera directa. Tiene drivers para el hardware y provee discos virtuales y acceso a la red a los invitados, a quienes denomina domU [27]. Para que la comunicación entre el dom0 y los domUs se realice se necesita de dos partes del controlador, las cuales son *BackendDriver y FrontendDriver*. Por otra parte el DomU o también conocido como dominio U, es el dominio sin privilegios, del cual pueden correr varios al mismo tiempo [26], pero no tiene acceso al hardware. La Figura 3.1. Muestra varios domUs o VM_n ejecutándose al mismo tiempo.

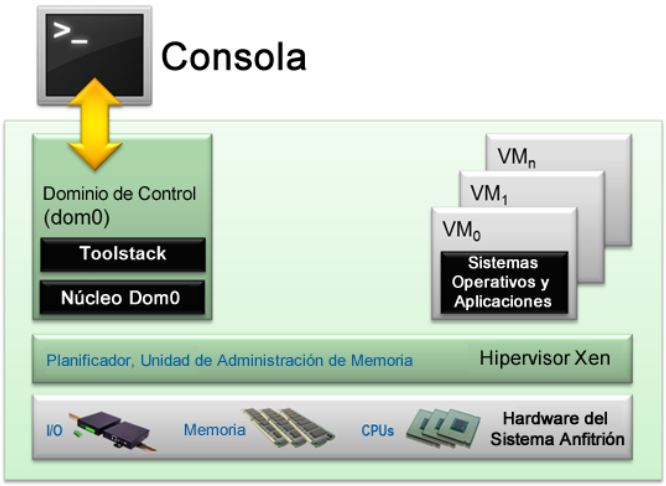


Figura 3.1. Arquitectura Xen [27]

Citrix Xen Server. Es una plataforma desarrollada por Citrix, que se basa en el proyecto Xen (que era GPL y fue absorbido por Citrix en el año 2007) y en la distribución Red Hat [28]. El hipervisor Xen Project es el componente principal de su arquitectura que permite proporcionar una abstracción estable y elástica de la infraestructura subyacente, como se muestra en la figura 3.2. Está diseñado para una administración eficiente de máquinas virtuales de Windows y Linux brindando una opción rentable de consolidación de servidores y continuidad del negocio. Hace uso de las tecnologías de hardware que existen en la actualidad como Intel VT® y AMD-V®. Al aplicar la técnica de paravirtualización evita las bajas penalizaciones de rendimiento, proporcionando alrededor del 2%, con los peores casos de rendimiento rondando el 8%, lo cual contrasta con las soluciones de emulación que habitualmente sufren penalizaciones del 20 % [29].

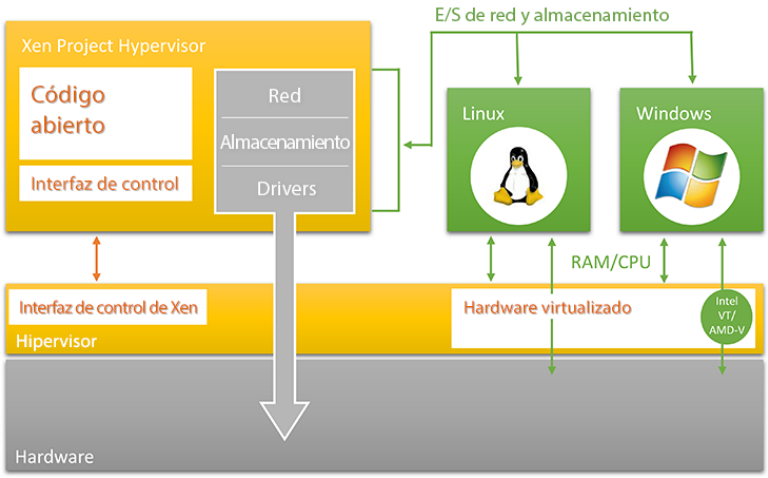


Figura 3.2. Diagrama de la arquitectura de Citrix Xen Server [30]

UML. Es una técnica de virtualización segura que permite ejecutar versiones y procesos Linux, de manera que se pueda ejecutar software defectuoso, experimentar con nuevos núcleos Linux o distribuciones, e indagar en la parte interna de Linux sin arriesgar la instalación principal de la misma [31]. Permite tener una máquina virtual con más recursos de hardware o software que la computadora física donde está instalada. Es usada especialmente para hospedar servidores virtuales, experimentar con nuevos núcleos y distribuciones y en la educación.

KVM. También conocida como Máquina Virtual basada en el núcleo. Es una solución de virtualización completa para Linux, creado y mantenido por Qumranet, que trabaja sobre procesadores x86 o x86_64 que contienen extensiones de virtualización Intel VT® o AMD-V®. KVM es un módulo del núcleo que fue incluido en Linux desde la versión 2.6.20 y trabaja con el núcleo por defecto [1].

En este modelo las máquinas virtuales son procesos normales (como se indica en la figura 3.3.), por eso la gestión de memoria y planificación de procesos son las estándar del sistema, a los que se añade un modo de ejecución adicional considerado como *invitado*, a partir de los modos de ejecución estándar de Linux los cuales son *usuario y núcleo*. Así, una máquina virtual tendrá tres modos de ejecución (véase figura 3.3). En cuanto a la implementación del sistema, KVM está formado por dos componentes. Uno de ellos es el Controlador de dispositivos, el cual gestiona el hardware de virtualización, accesible desde el dispositivo /dev/kvm. Y el otro es el Programa de usuario, que emula el hardware del PC, para ello usa una versión modificada de qemu que se encarga de reservar la memoria de la máquina virtual y llamamiento al controlador anterior para ejecutar código en modo invitado [7].

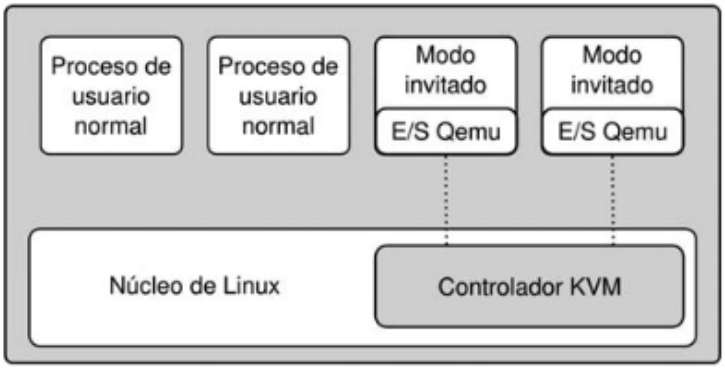


Figura 3.3. Arquitectura KVM [7]

VMWare vSphere. Es una plataforma de virtualización completa y de paravirtualización, líder del sector de infraestructura tecnológica. La arquitectura de hipervisor VMware vSphere ESXi™ (como se muestra en la

figura 3.4) proporciona una sólida capa de virtualización probada para entornos de producción y de alto rendimiento. Permite que varias máquinas virtuales compartan recursos de hardware con un rendimiento que puede igualar y en ocasiones superar al nativo [32].

Entre las aplicaciones que tiene vSphere, que proporcionan alta disponibilidad podemos mencionar a VMware vSphere vMotion, que permite la migración en caliente de máquinas virtuales entre servidores sin interrupción alguna para los usuarios ni pérdidas de servicio, eliminando así la necesidad de programar tiempo de inactividad de las aplicaciones para el mantenimiento de servidores. Otra de las aplicaciones que tiene es vSphere Storage vMotion, que permite la migración en caliente de discos de máquinas virtuales sin interrupción alguna para el usuario, eliminando la necesidad de programar tiempo de inactividad de las aplicaciones para realizar el mantenimiento programado o las migraciones del almacenamiento. También podemos mencionar a VMware High Availability (HA), la cual es una solución automatizada para el reinicio de todas las aplicaciones en cuestión de minutos en caso de un fallo de hardware o del sistema operativo. Por su parte, vSphere Fault Tolerance (FT) proporciona disponibilidad continua de todas las aplicaciones en caso de fallo de hardware, sin pérdida de datos ni tiempo de inactividad. Y finalmente, vSphere Data Recovery ofrece unas funciones de backup y recuperación sencillas, rentables y sin agentes de máquinas virtuales para entornos más pequeños [32]. De esta forma, vSphere proporciona una plataforma de virtualización de servidores que logra un aumento del 80% de la utilización

de los recursos de un servidor, un ahorro de hasta el 50 % en costes operativos y de capital, y una proporción de consolidación de servidores de 10:1 como mínimo [33].

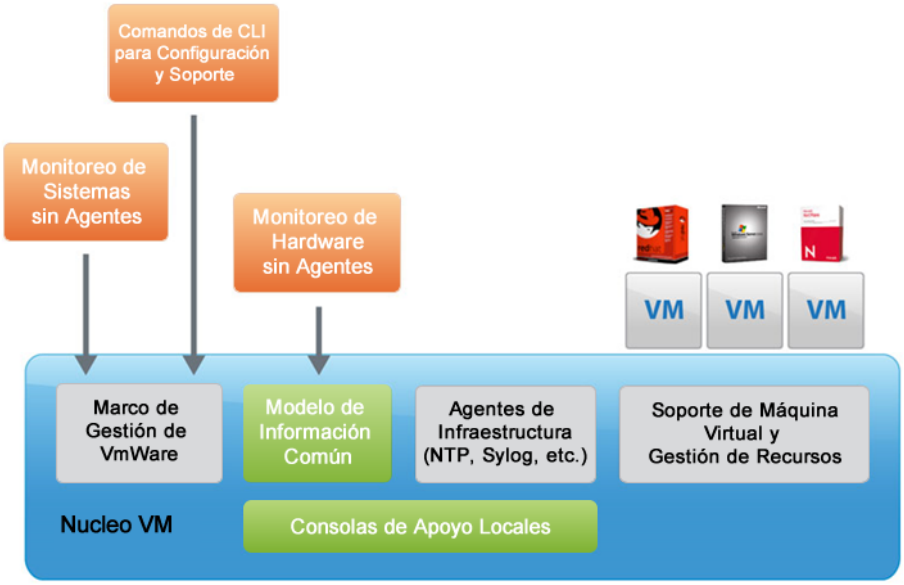


Figura 3.4. Diagrama de la arquitectura de Citrix Xen Server [33]

Hyper–V. Es una plataforma de Virtualización, que puede venir predefinido desde la versión de Windows server 2008 hasta Windows server 2012 R2 como un rol o como un producto autónomo llamado Hyper-V Server. Permite a través de la virtualización reducir costos debido a que se logra una mejor utilidad de los recursos. Se caracteriza porque se pueden ejecutar simultáneamente máquinas virtuales de arquitectura de 32 y 64 bits, además estas máquinas pueden tener uno o varios procesadores. Además, permite realizar instantáneas de las máquinas virtuales capturando el estado, los datos y la configuración de hardware de la misma

en ejecución. Proporciona compatibilidad con memoria de máquina virtual de gran tamaño y con la red de área local virtual [34].

Cada una de las soluciones que se mencionó en esta sección son aquellas que en la actualidad están siendo muy utilizadas debido a las ventajas que brindan. Todas ellas trabajan con la tecnología de virtualización completa o con Paravirtualización, o ambas a la vez. A continuación, en la Tabla 1 se resumirá las características de mayor relevancia de las soluciones que fueron parte de nuestro estudio.

Tabla 1. Resumen de Plataformas de Virtualización

Plataforma de Virtualización	**Tipo de Virtualización**	**Arquitectura CPU Anfitrión**	**Arquitectura CPU Invitado**	**Licencia**	**Costo**
Denali	Paravirtualización	X86	X86	Código Abierto	--
Xen	Virtualización Completa, Paravirtualización	x86, x86_64, ARM, IA-64, PowerPC	x86, x86_64, ARM, IA-64, PowerPC	Código Abierto	Gratuito, GPL

Tabla 1. Resumen de Plataformas de Virtualización (continuación)

Plataforma de Virtualización	**Tipo de Virtualización**	**Arquitectura CPU Anfitrión**	**Arquitectura CPU Invitado**	**Licencia**	**Costo**
Citrix Xen server	Virtualización Completa, Paravirtualización	x86, x86_64, ARM, IA-64, PowerPC	x86, x86_64, ARM, IA-64, PowerPC	Código Abierto.	Variable
UML (User Mode Linux)	Paravirtualización	x86, x86-64, PowerPC	x86, x86-64, PowerPC	Código Abierto	Gratuito, GPL v. 2
KVM	Virtualización Completa	x86, x86-64, IA-64, s390, PowerPC	x86, x86-64, IA-64, s390, PowerPC	Código abierto	Gratuito, GPL v. 2
KVM	Virtualización Completa	x86, x86-64, IA-64, s390, PowerPC	x86, x86-64, IA-64, s390, PowerPC	Código abierto	Gratuito, GPL v. 2

Tabla 1. Resumen de Plataformas de Virtualización (continuación)

Plataforma de Virtualización	Tipo de Virtualización	Arquitectura CPU Anfitrión	Arquitectura CPU Invitado	Licencia	Costo
VMWare	Virtualización Completa, Paravirtualiza-ción	x86, x86-64	x86, x86-64	Propietario	Variable
Hyper-V	Virtualización completa. Paravirtualización	x86-64 + virtualización de hardware asistido (Intel VT-x, AMD-V)	x86-64	Propietario	Variable

Para concluir con esta sección de análisis de plataformas de virtualización, es importante mencionar que el artículo “A Component-Based Performance Comparison of Four Hypervisors” [35] publicado en la IEEE (2013), muestra los resultados de pruebas de Benchmarking que se hizo a cuatro Hipervisores, los cuales son: Hyper-V 2008 R-2, KVM 2.6.32-279, WMware

vSphere 5.0 y XEN 4.1.2. Estos hipervisores fueron evaluados en el modo de Virtualización completa con apoyo de hardware.

Entre las pruebas de rendimiento realizadas están Bytemark, Ramspeed, Bonnie++, FileBench, Netperf, entre otras. Todas ellas se realizaron con el objetivo de medir y comparar el rendimiento tanto de memoria, CPU, instrucciones de entrada y salida, red, y otros parámetros. En este artículo se llegó a la conclusión que Xen obtuvo el rendimiento más bajo de los hipervisores, recalcando que estaba siendo probado en modo de virtualización completa y no en modo de paravirtualización para el cual fue creado. Además los otros hipervisores también presentaron penalizaciones. Por ejemplo en el caso de KVM, hay una sobrecarga en el uso de memoria cuando todos los núcleos del sistema están activos. Hyper-V por su lado presenta una disminución de la velocidad de los procesos cuando múltiples núcleos son dedicados a la ejecución de pequeñas y secuenciales lecturas y escrituras, y Xen también sufre un bajo rendimiento en la red.

Otro estudio de benchmark publicado en la IOPScience, titulado “A quantitative comparison betwen xen and kvm” (2010) [36], confirma que el rendimiento de KVM v.83 ante Xen v.3.2.1 en modo de virtualización completa es mayor. Sin embargo, también se hace comparaciones con Xen en modo de paravirtualización, el cual en su mayoría obtiene un mejor rendimiento que las otras dos soluciones, debido a que como se explicó en el capítulo anterior el sistema invitado es modificado de manera que no

dependa 100% del hipervisor, logrando un rendimiento muy cerca del nativo. En este estudio se realizaron pruebas como HEP-Spec06, Iperf y Bonnie++. De esta manera, se concluye que Xen en modo de paravirtualización sigue siendo la mejor solución de software libre.

3.3. Análisis de las diferentes soluciones de clúster.

En esta sección vamos a dar a conocer las soluciones de clúster que se acoplan mejor a la necesidad actual de brindar alta disponibilidad a los servicios web que brinda la FIEC. Este estudio nos permitirá ir comprendiendo mejor el diseño y la posterior implementación de la solución de alta disponibilidad, puesto que si en algún momento llegase a fallar el hardware del equipo donde se encuentra alojada la máquina virtual que simula al servidor web y de base de datos de la FIEC, el software de clúster y sus diferentes componentes permitirán que los servicios alojados en tal servidor sean migrados a otro equipo físico. A lo largo de este estudio nos encontraremos con la tecnología que ofrece Microsoft, la cual es muy completa ya que se encarga tanto de la comunicación entre los nodos del clúster, como de la administración del mismo. Sin embargo, también encontraremos soluciones que se encargan de una función específica dentro del clúster, pero que combinadas entre sí dan como resultado una tecnología de implementación de clúster robusta.

Clúster de conmutación por error de Microsoft. Es una tecnología de implementación de clúster que viene definida como un rol dentro del sistema operativo Windows Server. También es conocida como

Failover Clustering en sus siglas en inglés. Permite migrar la información de un nodo a otro nodo del clúster a través de la red con respecto a máquinas virtuales en caso de error, las cuales pueden ser configuradas con cierta prioridad de manera que se pueda controlar el orden en el que deberán iniciar.

Realiza monitoreos a nivel de aplicación y junto con el rol Hyper-v proveen de una solución para monitorear además las aplicaciones que se encuentran dentro de la máquina virtual, detectando fallas en los servicios que se están ejecutando y llevando a cabo acciones correctivas como reiniciarlos. Actúa en la movilidad de las máquinas virtuales incluyendo funciones como *live migration y quick migration* las cuales han estado presentes desde el sistema operativo Windows server 2008 r2 y ahora en la versión 2012 presenta la ventaja de permitir más de una operación simultánea entre los nodos del clúster. El *storage migration* es otra utilidad que permite mover el almacenamiento de una máquina virtual a otra ubicación. En el Windows server 2012 r2 existen mejoras de interoperabilidad entre el rol Clúster de conmutación por error con el *Active Directory,* ya que los nodos del clúster no requieren de comunicación con un controlador de dominio antes que el servicio sea iniciado [37].

Heartbeat. Actualmente se conoce con este nombre a la capa de mensajería de clúster. Sin embargo, hasta la versión 2.1.4 comprendió las funcionalidades de administrador de recursos locales, infraestructura “plumbing”, STONITH, agentes de recursos y administrador de recursos de

clúster, actualmente desarrollado de manera independiente bajo el nombre de Pacemaker. Es un demonio que provee los servicios de infraestructura de clúster a sus clientes, los cuales son comunicación y membresía. Necesita combinarse con un administrador de recursos de clúster (CRM) para mantener alta disponibilidad. Pacemaker es el CRM preferido para clústeres que se basan en Heartbeat [38].

Corosync Cluster Engine. Es un sistema de comunicación de grupo con características adicionales que permiten implementar alta disponibilidad dentro de las aplicaciones. Se derivó del proyecto OpenAIS y está bajo la licencia New BSD. Entre las características que ofrece se puede mencionar un modelo de comunicación de grupo con garantías de sincronización virtual que permite replicar el estado de las máquinas, un administrador de disponibilidad simple que reinicia los procesos de las aplicaciones en caso de fallo, configuración y estadísticas de base de datos en memoria que proporcionan la capacidad de establecer, recuperar, y recibir notificaciones de cambio de información y por último un sistema de quorum que notifica a las aplicaciones cuando el quorum se ha logrado o se ha perdido [39].

CMAN. Es un administrador de clúster basado en el núcleo, que se distribuye a todos los nodos. Consta de dos partes. La primera de ellas se encarga de administrar la conexión y maneja la membresía, mensajería, quorum, notificaciones de evento y transiciones. La segunda parte maneja los grupos de servicios. Es considerado como un sistema fundacional del cual dependen DLM, GFS, CLVM y Fence [40]. La suite de alta

disponibilidad de Red Hat Enterprise lo incluye como un Add-on, y también puede trabajar como un plugin junto a Corosync y Pacemaker, para ofrecer alta disponibilidad.

Mantiene el registro de la membresía del clúster mediante mensajes de control entre los nodos. Si un nodo no transmite un mensaje durante un tiempo preestablecido, CMAN lo removerá del clúster e informará lo sucedido a los otros componentes de la Infraestructura de clúster para que realicen las acciones necesarias. También mantiene un registro del quorum del clúster mediante el control de la cuenta de los nodos. Si más de la mitad de los nodos están activos el clúster tiene quorum, pero si tan solo la mitad o menos de la mitad están activos, el clúster no tiene quorum y la actividad se detendrá [18].

Pacemaker CRM. Es un software de código abierto que administra los recursos del clúster. Fue desarrollado a partir del 2004, por Red Hat y Novell, y recibe además apoyo de Linbit y la comunidad open source en general. Soporta varios stacks de mensajería como Heartbeat, Corosync y Cman y además varios escenarios de implementación, desde clústeres de 2 nodos hasta configuraciones de clústeres de 16 nodos. Reduce notablemente los costos de hardware. Supervisa además el sistema para manejar fallos tanto de hardware como de software, y en el caso que se llegue a dar algún fallo recuperará automáticamente la aplicación en una de las máquinas restantes del clúster, mediante algoritmos avanzados que determinarán cual es el lugar más apropiado [41].

El formato de configuración interna de Pacemaker es *XML*, el cual es ideal para las máquinas pero poco comprensible para los seres humanos. Debido a esto, desarrolladores de la comunidad han creado interfaces gráficas e interfaces de línea de comando que permitan la configuración del clúster sin tener que usar de manera directa *XML.* Entre las interfaces de línea de comando encontramos a *crmsh y a PC*, mientras que entre las interfaces visuales tenemos a *pygui, halcón, LCMC y PC* en esta modalidad [41]. Pacemaker puede trabajar con Heartbeat o con Corosync para ofrecer una solución completa clúster de alta disponibilidad.

RGManager. Al igual que Pacemaker es un administrador de recursos de clúster que gestiona y proporciona capacidades de conmutación por error para las colecciones de recursos del clúster llamados servicios, grupos de recursos, o árboles de recursos. Define, configura y supervisa los servicios del clúster. En caso de fallo de un nodo, rgmanager reubicará el servicio en otro nodo del clúster con una mínima interrupción del servicio. También puede restringir los servicios a determinados nodos, tales como la restricción de httpd a un grupo de nodos mientras mysql puede ser restringido a un conjunto separado de nodos. Existen varios procesos y agentes que se combinan para hacer el trabajo rgmanager. Entre ellas encontramos Dominios de conmutación por error, políticas de servicio, árboles de recursos, comportamiento de servicios operacionales, comportamiento de máquinas virtuales, acciones de recursos, secuencias de comandos de eventos [18].

Complemento de Alta Disponibilidad de Red Hat Cluster Suite. La suite de Red Hat Cluster es un conjunto integrado de componentes de software que pueden ser implementados en una variedad de configuraciones para adaptarse a las necesidades de rendimiento, alta disponibilidad, balanceo de carga, escalabilidad, intercambio de archivos y economía [18]. El componente de alta disponibilidad de Red Hat Cluster Suite está conformado por cman, el cual ya se dio a conocer de manera previa y debido a que no constituye un mecanismo de mensajería como Heartbeat o corosync, utiliza corosync como el demonio encargado de la mensajería en el clúster y que proporciona información importante como la membresía y quorum. El otro mayor componente es rgmanager, que también ya fue objeto de este estudio, el cual es un reemplazo totalmente funcional para Pacemaker y que trabaja exclusivamente con RedHat.

CAPÍTULO 4

4. DISEÑO, IMPLEMENTACIÓN Y PRUEBA DE LOS ESCENARIOS DE VIRTUALIZACIÓN Y DE IMPLEMENTACIÓN DE CLÚSTER

En este apartado se dará a conocer los diferentes diseños que se crearon para probar las tecnologías de virtualización y clúster, así como las pruebas que se hicieron de manera que el lector pueda darse cuenta cuál de las técnicas tuvo un mejor rendimiento y un mejor tiempo de respuesta ante un fallo, respectivamente.

Para llevar a cabo la implementación de las pruebas, se utilizaron tres computadoras de escritorio de última tecnología, las cuales a lo largo de este documento serán llamadas servidores físicos. Dos de estas computadoras son de la marca HP, modelo Compaq Pro 6300 Microtower, como se muestra en la Figura 4.1 y cuentan con un procesador Intel ® Core (TM) i7-3770 CPU @ 3.40GHz, 8.00 GB de RAM, una tarjeta de red Intel ® 82579LM Gigabit Network

Connection, a las cuales se les adaptó una tarjeta de red adicional a cada una, marca D-Link, modelo DGE-530T V.B1 Gigabit Ethernet. También se utilizó otra computadora de escritorio clon, con procesador Intel ® Core 2 Duo 2.53 GHz, 2 memoria RAM de 2GB cada una y un disco duro de 1 TB. Los dos primeros equipos son los que alojaron los servidores virtuales, quienes en el Diseño de Escenarios de Implementación de Clúster serán llamados Nodos.

Figura 4.1. Computadora de Escritorio HP [42]

También se hizo uso de un conmutador marca HP de 24 puertos como el que se muestra en la Figura 4.2., el cual permitió la interconexión de los servidores físicos para que éstos y los nodos alojados en los mismos tengan acceso a Internet. Finalmente se utilizó un conmutador de 8 puertos, el cual permitió que los tres servidores físicos se comuniquen entre sí a través de una red privada mediante la cual se monitoreaba el estado de cada uno de los nodos del clúster, de forma que si llegase a fallar uno de ellos, los recursos que se estuviesen ejecutando en el mismo migren al otro nodo para así tener continuidad operacional.

Figura 4.2. Conmutador HP [43]

4.1. Diseño de los escenarios de virtualización

En base al estudio previo que se hizo acerca de las diferentes soluciones de virtualización, inicialmente se concluyó que las tres opciones que deberían probarse eran VMware vSphere Hypervisor, el rol Hyper-V de Windows Server 2012 R2 y el hipervisor Xen, debido al alto rendimiento que presentaron al ser sometidos a pruebas de benchmark, según un artículo publicado en la IEEE [35], descartando a KVM Linux, puesto que a pesar del alto rendimiento que tiene, el hecho de no presentar un aislamiento total entre los servidores virtuales o máquinas virtuales, no cubría el requerimiento de seguridad que se deseaba para los servidores de la facultad. A pesar de que inicialmente se había decidido probar VMware vSphere Hypervisor 5.5.0 Update 1, luego se la descartó en vista que no existía soporte para la tarjeta de red del equipo de cómputo que se estaba utilizando [44], por lo que no se pudo instalar tal hipervisor. Debido a estas limitaciones, se optó por hacer las pruebas con Hyper-V instalada como rol en el Sistema Operativo Windows Server 2012 R2 y con el hipervisor Xen versión 3.17 en modo de virtualización completa con hardware asistido y

paravirtualización, cuyo sistema operativo anfitrión fue Fedora 20, como se muestra en la Tabla 2 y en la figura 4.3.

Tabla 2. Tecnologías de Virtualización a Probar

Tecnologías de Virtualización	Servidor Virtual
Rol Hyper-V de Windows Server 2012 R2.	CentOS 6.5.
Xen Server v. 3.17 sobre Fedora, técnica de virtualización completa con hardware asistido.	CentOS 6.5.
Xen Server v. 3.17 sobre Fedora, técnica de paravirtualización.	CentOS 6.5.

Estas pruebas se las realizó en las computadoras HP Compaq Pro. El sistema Operativo invitado para cada una de las opciones de virtualización, fue CentOS 6.5 como se muestra en la figura 4.3., con 1024 MB de memoria RAM, 1 CPU virtual y 20 GB de disco duro. No se probó con ningún otro sistema operativo debido a que el objetivo principal es implementar los servidores de la FIEC en máquinas virtuales, por lo tanto se mantuvieron los mismos parámetros, es decir, se mantuvo el sistema operativo que actualmente está instalado en los servidores de la facultad, aunque con una versión actualizada, puesto que la versión de CentOS sobre la cual se están ejecutando los servicios de la FIEC es la 5.4.

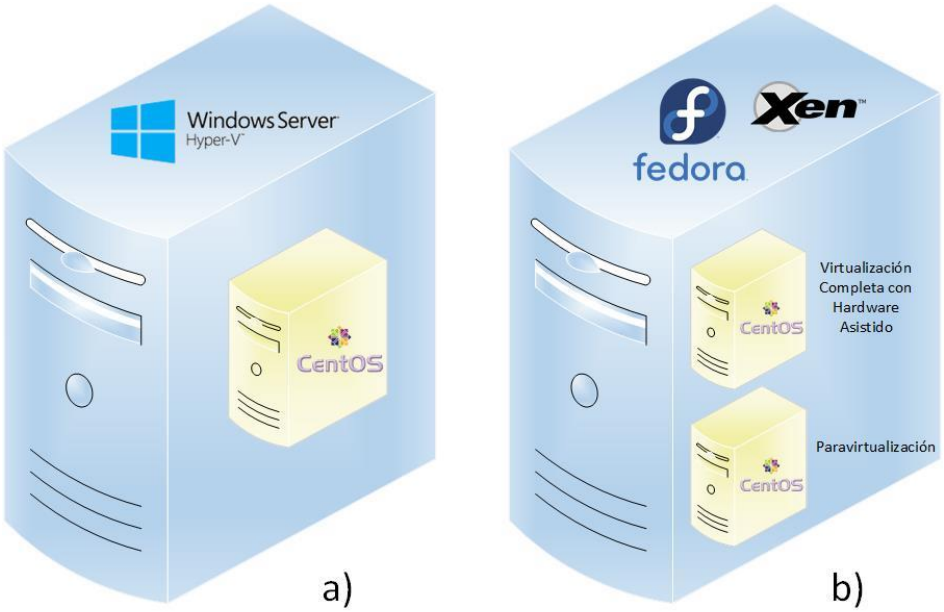


Figura 4.3. Diseño de escenarios de virtualización a) Hipervisor Hyper-V b) Hipervisor Xen

Para probar el rendimiento de las diferentes opciones de virtualización fue necesario generar carga de trabajo al servidor virtual, tal y como sucedería en un entorno de producción y realizar el respectivo monitoreo del consumo de recursos, para observar cuál tenía el mejor desempeño. La carga de trabajo al servidor virtual se la efectuó a través del uso del comando stress de Linux y la herramienta ab. El comando stress, es una herramienta que permite imponer carga de trabajo al procesador, memoria, operaciones de entrada/salida o al disco, durante un tiempo determinado. Por otra parte, la herramienta Apache Benchmarking tool, o también conocida como ab, es aquella que crea una simulación de carga a un servidor web mediante la solicitud simultánea de múltiples usuarios contra el servidor.

Los recursos que se monitorearon fueron CPU y memoria, debido a que la capacidad de los mismos son elementos fundamentales que determinan el alto o bajo rendimiento de un servidor. Para poder realizar la comparativa entre los servidores virtuales sobre las diferentes plataformas de virtualización, fue necesario obtener datos de las mediciones de estos recursos, en los momentos en los que se ejecutaban las técnicas de stress, para ello se hizo uso del comando sar, el cual es un comando que recoge, informa o guarda la información de la actividad del sistema. Es así como se planteó tres escenarios y en cada uno de ellos se realizaron dos pruebas de stress, como se indica en la tabla 3.

Tabla 3. Escenarios de Virtualización a probar

Tecnologías de Virtualización	**Servidor Virtual**	**Técnicas de Stress**
Rol Hyper-V de Windows Server 2012 R2.	CentOS 6.5.	stress
		ab
Xen Server 3.17, técnica de virtualización completa con hardware asistido.	CentOS 6.5.	stress
		ab
Xen Server 3.17, técnica de paravirtualización.	CentOS 6.5.	stress
		ab

4.2. Implementación y prueba de las diferentes plataformas de virtualización.

Como se estableció en el diseño de los escenarios de virtualización, se procedió a instalar sobre una de las computadoras de escritorio marca HP el Sistema operativo Windows Server 2012 r2, luego de lo cual fue necesario instalar el rol Hyper-V, que permitiría crear el servidor virtual de prueba [45]. En la otra computadora marca HP se instaló Fedora 20 como sistema operativo anfitrión y luego se instaló el hipervisor Xen Server v. 3.17 [46], además de instalar el paquete *libvirt* [47], el cual permitió a través de su herramienta *virt-manager* instalar y administrar los otros dos servidores virtuales de prueba mediante interfaz gráfica como se observa en la figura B-1 del anexo B, uno de los cuales se instaló en modo de virtualización completa con hardware asistido y el otro en modo de Paravirtualización, como se muestra en la figura B-2 del anexo B. Los tres servidores virtuales alojaron como sistema invitado a CentOS 6.5, del cual se hizo una instalación mínima, la cual constó del núcleo del sistema, un conjunto de herramientas básicas, paquetes indispensables para configurar las interfaces de red, herramientas básicas para administrar el sistema de archivos, un conjunto básico de políticas para SELinux, el gestor de paquetes yum y lo mínimo necesario para tener un sistema operativo funcional, lo cual constituye una buena práctica de seguridad puesto que posteriormente se instalará solo los paquetes que sean necesarios, además habrá menos servicios por los cuales preocuparse y las descargas de paquete durante las actualizaciones también serán menores.

Luego de haber instalado el sistema operativo CentOS sobre los servidores virtuales para cada una de las tecnologías de virtualización, se procedió a instalar el software respectivo para la ejecución de las pruebas de rendimiento. Para ello, fue menester instalar las herramientas que generarían carga de trabajo: stress y el servidor web Apache que incluye el comando de benchmarking ab y la herramienta de monitoreo sar. Además fue necesario definir el número de repeticiones de las pruebas, para lo cual se utilizó la fórmula del tamaño de la muestra, que se detalla en el Anexo A, y cuyo resultado fue 96. En la figura 4.4 se pueden observar los comandos stress (a) y ab (b) que se utilizaron para generar la carga de trabajo a los servidores virtuales con sus respectivos parámetros. En el anexo C, el lector encontrará la definición de cada parámetro de las herramientas de carga de trabajo usadas en este proyecto.

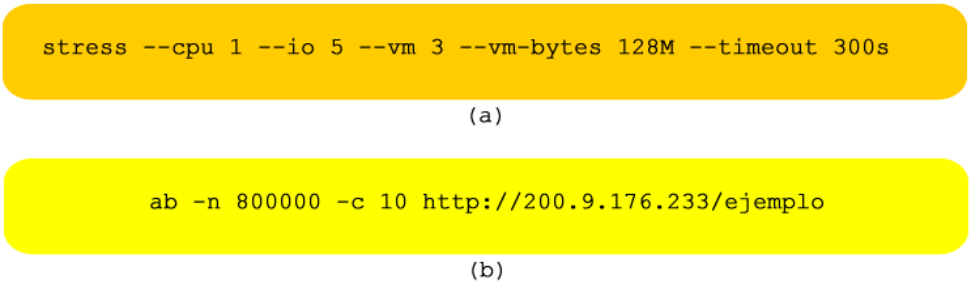


Figura 4.4. Herramientas de carga de trabajo (a) stress (b) ab

Por otra parte, en la figura 4.5 se muestran dos ejemplos de cómo debió definirse el comando sar, tanto para el monitoreo de cpu (a), como para el monitoreo de memoria ram(b). Asimismo en el anexo C, se encuentra

información de los parámetros que se requirieron para la ejecución del mismo.

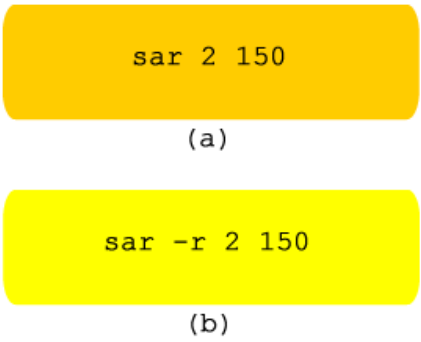


Figura 4.5. Herramienta de monitoreo sar (a) para cpu y (b) para ram

Dentro del directorio /root/Documentos de cada uno de los servidores virtuales se creó una carpeta con el nombre result, donde se almacenaron los resultados de las pruebas de carga de trabajo. Habiendo ya definido cuáles serían los comandos a usar, tanto para generar la carga de trabajo, como para monitorear el uso de los recursos durante la actividad de carga, lo que se procedió a hacer primero fue ejecutar el comando de monitoreo cuyo resultado se iba guardando en un fichero dentro de la carpeta result, e inmediatamente enviar a ejecutar los comandos de carga de trabajo. Para el caso de stress, fue necesario abrir otro terminal dentro del mismo servidor virtual y ejecutar tal comando. Sin embargo, para la ejecución de ab, se utilizó un equipo cliente desde el cual se hicieron las peticiones al servidor virtual, como se indica en la figura B-3 del anexo B. El equipo cliente tenía instalado al igual que los servidores virtuales, el sistema operativo CentOS 6.5 y el servidor apache.

De esta manera, se iría guardando en el fichero la actividad de los recursos durante el tiempo de carga de trabajo. Finalmente, se procedió a tomar los 96 datos definidos por la fórmula del tamaño de la muestra de los datos recolectados en los ficheros, obteniendo así los resultados del rendimiento de memoria RAM y CPU que se muestran en las tablas 8, 9 y 10 en la sección de análisis de resultados de los diferentes escenarios de virtualización y soluciones de clúster.

4.3. Diseño de los escenarios de implementación de clúster.

El diseño que se especificará en esta sección, utilizado para probar las diferentes soluciones de clúster estudiadas con anterioridad corresponde a un clúster de conmutación de error conformado por dos nodos o servidores virtuales, los mismos que se instalaron sobre las dos computadoras marca HP, denominadas servidores físicos. Para la instalación de los servidores virtuales se utilizó la tecnología de virtualización Xen en el modo de paravirtualización y el sistema operativo de los nodos fue CentOS 6.5. El motivo por el cual se utilizó esta técnica de virtualización está descrito en el siguiente capítulo en la primera sección que corresponde al Análisis de los resultados de las diferentes soluciones de virtualización y clúster. El clúster de conmutación de error estuvo conformado por un nodo activo y un nodo pasivo, de manera que solo el nodo en estado activo provea los servicios correspondientes, y en caso de fallo del mismo, los recursos o servicios se migren al nodo pasivo provocando así que haya continuidad operacional.

Es importante recalcar que los nodos se encontraron en diferentes servidores físicos, como se describe en la figura 4.6, respetando así el diseño de alta disponibilidad debido a que si uno de los servidores físicos llegase a fallar, los nodos o servidores virtuales del otro servidor físico entrarían en funcionamiento.

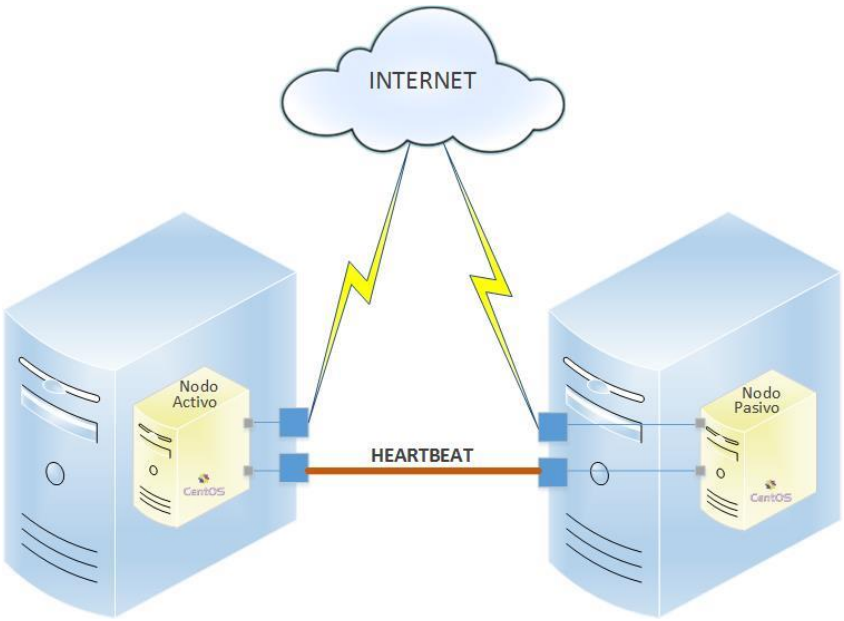


Figura 4.6. Clúster virtual de dos nodos

Para probar las soluciones de clúster se crearon dos tipos de clúster, uno de ellos conformados por los servidores virtuales web y el otro conformado por los servidores virtuales de base de datos, tal y como se indica en la tabla 4.

Tabla 4. Escenarios de soluciones de clúster a probar

Soluciones de Clúster	Clúster
Pacemaker + Corosync + CMAN	Clúster de servidores web
	Clúster de servidores de bases de datos
Complemento de alta disponibilidad de RedHat Cluster	Clúster de servidores web
	Clúster de servidores de bases de datos

Para que exista comunicación entre los servidores virtuales fue necesario crear dos *puentes de red* en cada uno de los servidores físicos. Uno de los puentes permitió la comunicación entre los nodos para las tareas de monitoreo a través de la técnica de heartbeat. Mientras que para la comunicación de los nodos con Internet, la configuración fue diferente para el clúster de servidores web y el clúster de servidores de bases de datos. Debido a que en el primero sí se usó un puente para que los nodos tengan acceso a internet, mientras que para el otro con la herramienta libvirt, se configuró un NAT, de manera que los nodos de dicho clúster tendrían IPs privadas, proporcionadas por DHCP que les permitiría tener acceso a Internet. La tabla 5 muestra el direccionamiento de los nodos de los clústeres para cada una de las máquinas virtuales.

Tabla 5. Direccionamiento de los Nodos de los clústeres

Tipos de Clúster	Nombre de Nodo	IP Heartbeat (br1)	IP Internet (br2)
Clúster de Servidores Web	Nodo1	192.168.5.11/24	200.9.176.233/25
	Nodo2	192.168.5.21/24	200.9.176.234/25
Clúster de Servidores de Bases de Datos	Nodo1	192.168.5.12/24	DHCP (NAT)
	Nodo2	192.168.5.22/24	DHCP (NAT)

Para poder elegir las soluciones que se probarían previo a la definición de cuál sería la solución de clúster final, se debió tomar en cuenta varios aspectos, entre los cuales está el sistema operativo de los nodos que conformarían el clúster, el cual en este caso es CentOS 6.5, por lo que en una primera instancia se descartó la solución de Windows Server 2012 R2 con su complemento de Clúster de conmutación por error de Microsoft, debido a que esta solución de clúster solo se puede implementar sobre sistemas Microsoft Windows. De esta manera se decidió que las dos soluciones a probar serían el conjunto conformado por Pacemaker, Corosync y CMAN, y el complemento de alta disponibilidad de RedHat Cluster.

La combinación de Pacemaker y Corosync es conocido como un “stack”. Por sí solo, este stack puede conformar una solución de alta disponibilidad completa, sin embargo en RHEL6 y por consiguiente en sus derivados como lo es CentOS 6, el “stack” soportado se basa en CMAN. De manera

que el “stack” que se usó para esta primera solución estuvo conformado por Pacemaker, CMAN y Corosync como se lo advirtió inicialmente. CMAN contiene *APIs* que pueden ser usados por Pacemaker para obtener información de membresía y quorum, y asimismo obtiene está información al trabajar con Corosync por debajo, como un demonio que CMAN administra, quien ya no se configura con el fichero corosync.conf sino con el fichero cluster.conf, el cual es similar en todos los nodos que conforman el clúster a diferencia del fichero corosync.conf que suele ser distinto en cada uno de los nodos [49]. Para poder realizar tareas de configuración y monitoreo de esta solución de clúster se procedió a trabajar con la interfaz de línea de comando shell crmsh y además con la herramienta de configuración pcs. Debido a que se trataba de un clúster de dos nodos, se configuró que se ignore el quorum, de manera que al fallar uno de los nodos el otro continúe funcionando. También se especificó el orden de ejecución de los servicios al arrancar, debido a que había servicios que dependían de otros y por lo tanto el orden en el que se ejecutasen era muy importante. Fue menester configurar que todos los servicios se ejecuten en el mismo nodo y que solo en el caso de que el nodo activo falle, todo el grupo de recursos se migren por completo al otro nodo. Gracias a la posibilidad que Pacemaker da para configurar el tiempo de monitoreo de cada recurso, se pudo especificar esta opción y además definir que en el caso de que un recurso falle, Pacemaker se encargue primero de reiniciarlo dentro del mismo nodo.

La segunda solución que se probó en el desarrollo de este proyecto fue el complemento de alta disponibilidad de Red Hat Cluster, el cual ya se explicó en el capítulo anterior en la sección de análisis de las diferentes soluciones de clúster. Para la administración del clúster de Red Hat, se trabajó con la interfaz gráfica *Conga*, para lo cual fue necesario instalar el agente *Ricci* en cada uno de los nodos y se seleccionó una computadora donde se instaló el servidor *luci*, desde la cual se pudiera configurar el clúster y sus recursos de manera remota. Al igual que la otra solución debió definirse una regla para que en el caso de que uno de los nodos falle, el otro nodo activo continúe brindando los servicios, a pesar de ya no haber quorum. Además, la interfaz gráfica permitió de una manera muy sencilla agregar los recursos que se requirieron para el clúster, formar conjuntos de los mismos denominados servicios y crear árboles de recursos. Estos últimos, como su nombre lo indica permitieron que los recursos se estructuren en forma de árbol y tengan dependencia de padre-hijo y relaciones de herencia dentro de cada subárbol [50], lo cual fue muy importante al momento de definir el orden en el que arrancarían cada uno de los recursos. También se pudo especificar las políticas de servicios a través de Conga, de tal forma que se pudiese administrar el inicio de servicio de RGManager y las políticas de recuperación.

Los recursos que se crearon en el clúster de servidores de bases de datos fueron VirtualIP y BaseDatos, mientras que en el clúster de servidores web se crearon los recursos VirtualIP, Tomcat6 y Httpd, para cada una de las soluciones de clúster. El recurso VirtualIP, fue el encargado de administrar

la IP virtual a través de la cual los clientes podían acceder al clúster y en caso de fallo ésta migraría al nodo en buen estado, de tal manera que el proceso de recuperación ante un fallo sea transparente y el cliente no lo note, puesto que siempre se usaría la misma IP, como si de un mismo equipo se tratara. Los recursos BaseDatos, Tomcat6 y Httpd administrarían los servicios mysql, tomcat y apache, respectivamente.

Una vez definido el hardware sobre el que operarían las máquinas virtuales que conformarían los clústeres, el sistema operativo de los nodos, las soluciones de clúster a probar y los recursos a implementar, fue importante definir el parámetro que se utilizaría para comparar ambas soluciones. Fue así como se decidió que el parámetro de comparación sería el tiempo de respuesta ante un fallo. Una de las técnicas de inyección de fallo consistió en matar los procesos de algunos servicios que se estaban ejecutando en cada uno de los clústeres y monitorear el tiempo que cada uno de los administradores de recursos se demoraban en detectar que el servicio no se estaba ejecutando y levantarlo nuevamente. La segunda técnica de inyección de fallo consistió en apagar súbitamente el nodo que se encontraba activo, y medir el tiempo que los servicios demoraban en restablecerse en el otro nodo del clúster. La tercera técnica, la cual no era precisamente una técnica de inyección de fallo, consistió en mover los recursos manualmente del nodo activo al nodo pasivo, mediante el uso de comandos, lo cual suele darse en tareas de mantenimiento del clúster. Aunque, ésta última no necesariamente constituía una técnica de inyección de fallo permitió tener una idea de cuánto demorarían migrar los recursos,

en el momento en que se requiriera hacer un mantenimiento. Para conocer los tiempos de respuesta, fue necesario obtener la información almacenada en los *logs* de los servicios a los cuales se les inyectarían fallos y también en los logs de los administradores de recursos de clúster como rgmanager, o del log de la capa de mensajería corosync. De esta forma los escenarios para las técnicas de inyección de fallo quedaron como se muestra en la tabla 6.

Tabla 6. Escenarios para las técnicas de inyección de fallo

Soluciones de Clúster	**Clúster**	**Técnicas de Inyección de Fallo**
Pacemaker + Corosync + CMAN	Clúster de servidores Web	Fallo a nivel de servicios
		Fallo a nivel de nodos
		Mover recursos manualmente
	Clúster de servidores de bases de datos	Fallo a nivel de servicios
		Fallo a nivel de nodos
		Mover recursos manualmente

Tabla 6. Escenarios para las técnicas de inyección de fallo

(continuación)

Soluciones de Clúster	Clúster	Técnicas de Inyección de Fallo
Complemento de alta disponibilidad de RedHat Cluster	Clúster de servidores web	Fallo a nivel de servicios
		Fallo a nivel de nodos
		Mover recursos manualmente
	Clúster de servidores de bases de datos	Fallo a nivel de servicios
		Fallo a nivel de nodos
		Mover recursos manualmente

4.4. Implementación y Prueba de las diferentes soluciones de clúster usando técnicas de inyección de fallos.

Siguiendo el diseño establecido previamente, se crearon las máquinas virtuales que conformarían los clústeres, cada una con el sistema operativo CentOS 6.5, con la instalación mínima y con las mismas características de hardware virtual que se utilizó para hacer las pruebas de virtualización, cuyos nombres se describen en la Tabla 7, para cada una de las soluciones de clúster. Previo a la instalación de los paquetes para ambas soluciones, se debió realizar algunas configuraciones en los nodos de manera que el clúster pueda funcionar sin inconvenientes. Estas configuraciones fueron:

Deshabilitar el servicio *NetworkManager* puesto que mientras éste estuviese habilitado el servicio CMAN no podría arrancar, configurar el módulo de seguridad *SELinux* en modo deshabilitado, agregar los nombres de los nodos junto a sus IPs en el archivo hosts ubicado en el directorio etc/, habilitar el servicio *ntpd* y para facilidad de las pruebas se prefirió deshabilitar el servicio iptables. Toda esta configuración se encuentra de manera detallada en el anexo D, en la sección Configuración previa a la instalación de las soluciones de clúster.

Tabla 7. Soluciones de clúster

Soluciones de Clúster	Tipos de clúster	Nombre de Nodos
Pacemaker, Corosync y CMAN	Clúster de servidores web	webPacemaker
		webPacemaker2
	Clúster de servidores de bases de datos	basePacemaker
		basePacemaker2
Complemento de alta disponibilidad de RedHat	Clúster de servidores web	webRGManager
		webRGManager2
	Clúster de servidores de bases de datos	baseRGManager
		baseRGManager2

Luego de haber hecho estas configuraciones se procedió a instalar los paquetes Pacemaker, corosync, cman, pcs, ccs y resource-agents, para la implementación de la primera solución de clúster. Teniendo ya todos los paquetes instalados, se creó el fichero cluster.conf en uno de los nodos, donde se especificó el nombre del clúster y los nodos que formarían parte del mismo y un método de fencing. Luego se copió el fichero cluster.conf

en el otro nodo, y se procedió a configurar el fichero cman en todos los nodos, para que en el caso de no haber quorum, de todos modos se inicie el clúster y posteriormente se iniciaron los servicios cman y pacemaker. Después se instaló la interfaz de línea de comando shell crmsh. Las opciones adicionales que se configuraron para esta solución fueron: Deshabilitar stonith, ignorar el quorum y configurar la propiedad resource-stickiness, la cual al darle el valor INFINITY indicaba que los recursos deberían quedarse en el nodo que se encontraban y solo abandonarían tal nodo ante un fallo del mismo. Finalmente se procedió a crear los recursos para cada uno de los clústeres, en el caso del clúster de Base de Datos se crearon los recursos VirtualIP y BaseDatos y en el caso del clúster Web se crearon los recursos VirtualIP, Tomcat6 y Httpd, con las respectivas restricciones que harían que todos los recursos se ejecuten en el mismo nodo y que solo se trasladen al otro nodo en caso de fallo, además de que se ejecuten en el orden adecuado. Para configurar estas restricciones, se utilizaron los comandos pcs constraint colocation y pcs constraint order, respectivamente. Esta parte de la implementación, el lector la podrá encontrar descrita detalladamente en el Anexo D, correspondiente a la sección Creación de Clúster con Pacemaker, CMAN y Corosync.

Por otro lado, para la implementación del complemento de alta disponibilidad de RedHat, se instalaron los paquetes “High Availability” y ricii en los nodos del clúster, y los paquetes “High Availability Management” y ricci en la máquina de administración del clúster. Luego se procedió a crear manualmente el fichero cluster.conf en el directorio /etc/cluster, el cual

contuvo inicialmente el nombre del clúster, la versión y los nodos que lo conformaban. Debido a que se trataba de un clúster de dos nodos, fue necesario especificar la siguiente instrucción: <cman two_node="1" expected_votes="1"> </cman> en ese fichero, de manera que si uno de los nodos llegase a fallar el otro nodo continúe dando los servicios correspondientes a pesar de no haber quorum. Después se deshabilitó el servicio iptables y se habilitaron los servicios ricci, cman, rgmanager y modclusterd en los nodos para que se inicien en el arranque y finalmente se los inició. Se configuró además una contraseña para ricci, la cual sería usada al momento unir los nodos a la interfaz gráfica de administración Conga y administrarlos. Además, en la máquina virtual de administración se iniciaron los servicios luci y ricci, luego de lo cual se pudo ingresar a la interfaz gráfica, a través del navegador Mozilla Firefox. Una vez en la interfaz gráfica de administración, se procedió a unir los nodos que conformaban los dos clústeres que posteriormente serían sometidos a prueba, y se agregaron los recursos IP Virtual y mysql para el clúster de Servidores de Bases de Datos y los recursos IP Virtual, apache y tomcat para el clúster de servidores Web. No se agregó un método de fencing, ya que las opciones que daba la interfaz gráfica hacían referencia a hardware especial que viene en equipos especializados para servidores, y en vista que las pruebas se estaban realizando en computadores de escritorio, no se contaba con tal hardware. Todo el proceso de instalación de la segunda solución de clúster se encuentra descrita detalladamente en el anexo D, en la sección Creación del clúster con el complemento de alta disponibilidad

de RedHat. Además en la sección B, correspondiente a las Figuras, el lector encontrará desde la Figura B-4 hasta la Figura B-8, las pantallas de la interfaz de administración del clúster donde se podía agregar los nodos, crear los dominios de conmutación del clúster, agregar los recursos, y agrupar tales recursos en los denominados servicios.

Luego de haber implementado las dos soluciones de clúster, fue necesario inyectar las técnicas de fallo que se nombraron en el diseño y medir el tiempo de respuesta de cada una de ellas. Para la toma de los datos del tiempo de recuperación de las soluciones de clúster ante la técnica de inyección de fallos a nivel de nodos se usó el comando virsh destroy <NombreMaquinaVirtual>, que permitió apagar súbitamente la máquina virtual, y luego se procedió a tomar el tiempo de recuperación de los servicios en el otro nodo de la información almacenada en los logs de los servicios que se estaban ejecutando en el clúster. Para tomar los datos del tiempo de recuperación de la técnica de inyección de fallos a nivel de servicios, se creó un script para cada una de las soluciones de clúster y para cada uno de los servicios a monitorear. Estos scripts se muestran en el anexo E, y difieren dependiendo del servicio que se requería detener y del administrador de recursos de clúster. Siendo así que para matar el servicio mysql se utilizó el comando kill, mientras que para matar el servicio tomcat en Pacemaker fue necesario ir al directorio donde se almacenaban los archivos de configuración de tomcat y ejecutar el script de apagado, y en RGManager bastó con ejecutar el comando service tomcat stop. La diferencia para apagar este servicio en cada una de las soluciones, radicó

básicamente en la forma en la que se instaló ese servidor. Puesto que para Pacemaker se descomprimió el archivo en la página oficial de tomcat, pero debido a que no contaba con un archivo que requería RGManager para manejar este servicio, para esta solución se lo tuvo que instalar a través de un repositorio. Los tiempos de respuesta se obtuvieron de los logs que suelen almacenar los servicios que se estaban probando. En el caso de los servidores de base de datos, se obtuvo la información del log de mysql el cual se encontraba en el directorio /var/lib/mysqld.log. Finalmente para tomar los datos del tiempo que se demoraban en migrar los recursos del nodo activo o al nodo pasivo, al hacer la operación de migración de forma manual, se utilizó unos scripts que se muestran en el anexo E, y el resultado de los mismos se comparó con el tiempo en el que se recuperaban los servicios en el nodo pasivo. Los datos y el análisis de las mismas se encuentran en el siguiente capítulo, en la sección de Análisis de los resultados de los diferentes escenarios de virtualización y soluciones de clúster.

CAPÍTULO 5

5. ANÁLISIS DE LOS RESULTADOS E IMPLEMENTACIÓN DE LA SOLUCIÓN

Luego de haber implementado y probado varias soluciones tanto para la tecnología de virtualización como para clúster, en este capítulo se mostrará el análisis que se realizó de cada una de ellas en base a los resultados obtenidos de las pruebas de rendimiento y de las técnicas de inyección de fallo. Este análisis permitió escoger las mejores soluciones tanto de virtualización como de clúster, para que formen parte de la solución de alta disponibilidad final. Además, en este capítulo el lector podrá apreciar el diseño y la implementación de la solución final de alta disponibilidad basada en virtualización y clúster para los Servidores de la FIEC.

5.1. Análisis de resultados de los diferentes escenarios de virtualización y soluciones de clúster

A lo largo de esta sección, el lector encontrará el análisis que se hizo en base a los resultados obtenidos al realizar las pruebas de carga de trabajo en las tres plataformas de virtualización y las técnicas de inyección de fallo en las dos soluciones de clúster. Se mostrarán las tablas de los resultados de las diferentes pruebas y además gráficos que ilustren el rendimiento y el tiempo de respuesta para cada una de las técnicas implementadas. Luego de leer el análisis, el lector podrá comprender las razones por las cuales se eligieron la plataforma de virtualización y la solución de clúster que finalmente integraron la solución de alta disponibilidad para los servidores de la FIEC.

Como se explicó en el capítulo anterior, los métodos para generar carga de trabajo en las tres plataformas de virtualización, fueron ab y stress. Los elementos que se monitorearon durante estas pruebas fueron la memoria RAM y el procesador. De ahí que los resultados del promedio de consumo de memoria RAM, como se observan en la tabla 8 y en la figura 5.1, muestran que al generar carga de trabajo con la prueba ab, la plataforma de virtualización Xen en modo de paravirtualización presentó el menor consumo de memoria RAM con un 56,03 %, seguida del hipervisor Hyper–V con un 59,85 %, y posterior a ellos, Xen en el modo de virtualización completa presentó un mayor consumo de memoria RAM con un 93,19%, el cual fue un valor extremadamente grande comparado con los otros dos que diferían en tan solo un 3,82%. Este primer análisis nos permitió concluir que

la plataforma de virtualización Xen en modo de virtualización completa no es recomendada para alojar servidores web, debido al alto consumo de memoria RAM que presenta al momento de responder las solicitudes web. Por otro lado, al realizar la prueba stress en las tres plataformas de virtualización, se observó un consumo similar en todos los hipervisores, siendo el hipervisor Xen quien presentó el menor consumo de memoria RAM con valores de 78,04 % y 79,55% para los modos de virtualización completa y paravirtualización, respectivamente. Hyper-V, por su lado presentó un consumo de 86,51% de memoria RAM en la prueba stress.

Tabla 8. Resultados del promedio de consumo de memoria RAM

	Pruebas de stress	
Plataforma de virtualización	**ab**	**stress**
Virtualización completa – Hyper-V	59,85 %	86,51 %
Virtualización completa - Xen	93,19 %	78,04 %
Paravirtualización - Xen	56,03 %	79,55 %

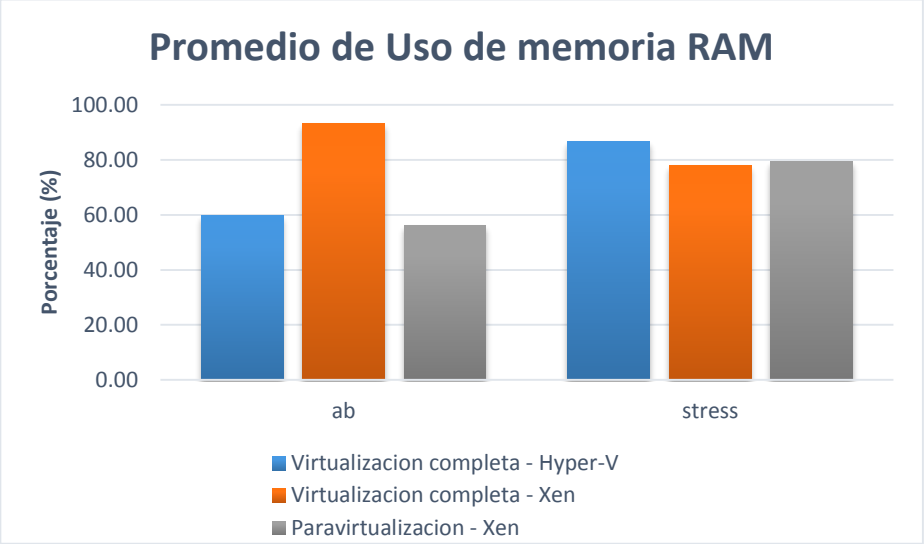


Figura 5.1. Promedio de Uso de memoria RAM

Como se mencionó anteriormente otro de los elementos, cuyo comportamiento fue monitoreado al momento de realizar las pruebas de carga de trabajo fue el procesador. El comando sar usado para el monitoreo del consumo de CPU, permitió conocer el porcentaje que se consumía de este recurso para los modos de ejecución de sistema y de usuario. En la tabla 9 se muestra el consumo de CPU en el modo de ejecución de sistema. Al realizar la prueba ab, Hyper-V presentó el menor consumo de CPU por parte del sistema, con un 40,20%, seguido de Xen en modo de paravirtualización con un 46,81% y Xen en modo de virtualización completa con el mayor consumo de CPU con un 54,89%. Para la prueba stress en cambio, el Hipervisor Xen en modo de virtualización completa presentó el menor rendimiento con un 80,93% seguido del mismo hipervisor pero en modo de paravirtualización con un 83,72%, y finalmente el que consumió más de este recurso, por parte del sistema fue Hyper-V con un 84,67%.

Tabla 9. Resultados del promedio de consumo de CPU por el sistema

	Pruebas de stress	
Plataforma de virtualización	**ab**	**stress**
Virtualización completa – Hyper-V	40,20 %	84,67 %
Virtualización completa - Xen	54,89 %	80,93 %
Paravirtualización - Xen	46,81 %	83,72 %

En la tabla 10, se muestran los resultados del promedio de consumo de CPU, en el modo de ejecución usuario. Con la prueba ab, Xen mostró menor consumo de CPU para las técnicas de Virtualización Completa y Paravirtualización, con 19,24% y 19,28%, respectivamente. Hyper-V presentó un mayor consumo de CPU en el modo de ejecución usuario con un 21,55%. En la prueba stress Hyper-V fue el que menos consumió CPU con un 15,33% seguido de Xen en modo de paravirtualización y Xen en modo de virtualización completa con un 18,68%. La figura 5.2 muestra el consumo de CPU durante cada una de las pruebas de estrés, para cada una de las plataformas de virtualización. Los detalles de los resultados de las pruebas se pueden observar con mayor detalle en el anexo F.

Tabla 10. Resultados del promedio de consumo de CPU por el usuario

	Pruebas de stress	
Plataforma de virtualización	**ab**	**stress**
Virtualización completa - Hyper-V	21,55 %	15,33 %
Virtualización completa - Xen	19,24 %	18,68 %
Paravirtualización - Xen	19,28 %	16,28 %

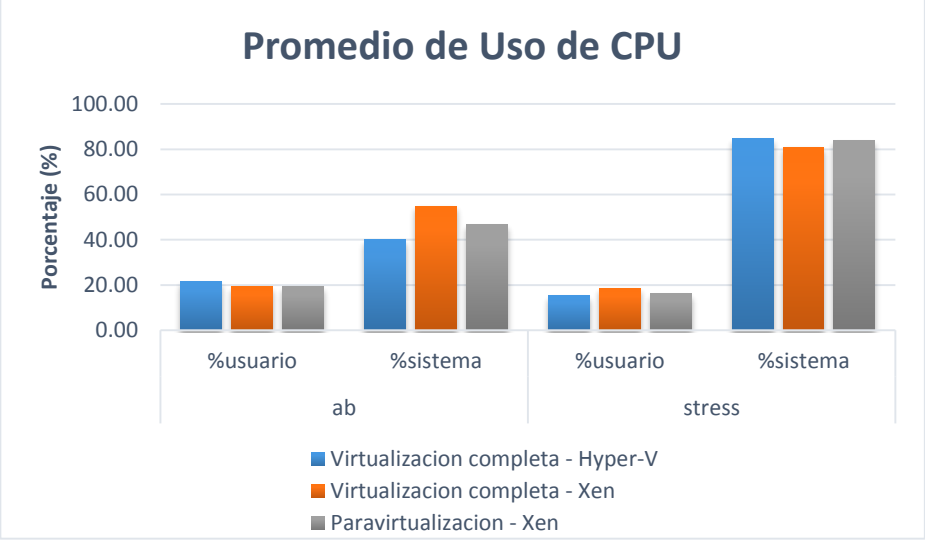


Figura 5.2. Promedio de Uso de CPU

Por otra parte, los resultados que se obtuvieron de las dos soluciones de clúster que se probaron, giraron en torno a las técnicas de inyección de fallo que se aplicaron para cada una de las soluciones. Para hacer la comparación nos referiremos a estas dos soluciones con el nombre de los administradores de recursos de clúster, es decir Pacemaker y RGManager. Como se mencionó en el capítulo anterior, las técnicas de inyección de fallo

se dieron a nivel de servicios y a nivel de fallo de los nodos de los clústeres. Las tablas 11 y 12, al igual que las figuras 5.3 y 5.4 muestran los resultados obtenidos al aplicar tales técnicas, para el clúster de servidores de base de datos y el clúster de servidores web, respectivamente. Los resultados de la técnica de inyección de fallo a nivel de servicios para el clúster de Servidores de bases de datos, muestran que el tiempo de respuesta de Pacemaker fue de 3,67 segundos, con un tiempo de monitoreo de 10 segundos para cada recurso, mientras que el tiempo de respuesta de RGManager fue de 27,94 segundos. Así mismo, para el clúster de servidores web, el tiempo de respuesta de Pacemaker fue de 6,33 segundos, con un tiempo de monitoreo de 10 segundos, mientras que RGManager presentó un tiempo de respuesta de 28,90 segundos. Cuando se aplicó la técnica de inyección de fallo a nivel de nodos, Pacemaker reaccionó favorablemente, dando un tiempo de recuperación ante el fallo de 10,17 segundos en el clúster de servidores de bases de datos, y de 9,43 segundos en el clúster de servidores web. Sin embargo al momento de aplicar esta técnica de inyección de fallo, RGManager no reestableció los servicios en el otro nodo que se encontraba activo en el clúster, a pesar de hacer todas las configuraciones que se requerían para que en el momento en el que falle el nodo, haya la conmutación y los servicios se inicien en el otro nodo. Debido a esta limitación, no se pudo obtener los tiempos de respuesta de inyección de fallo a nivel de nodos, por parte de RGManager. Sin embargo, manualmente se hizo el cambio de los recursos a través de los comandos que se especificaron en el capítulo anterior, obteniendo de

esta forma que Pacemaker demoraba 3,58 segundos en promedio en para migrar los recursos del nodo activo al otro nodo, mientras que RGManager demoró 20,14 segundos en migrarlos, para el clúster de bases de datos. Los tiempos de respuesta al migrar los recursos en el clúster Web fueron 2,60 segundos con Pacemaker y 20,42 segundos en RGManager.

Tabla 11. Tiempos de recuperación en el clúster de servidores de bases de datos

	CRM	
Técnicas de inyección de fallo	**Pacemaker**	**RGManager**
Fallo de Servicios	3,67 s	27,94 s
Fallo de Nodos	10,17 s	--
Migrar recursos por Mantenimiento	3,58 s	20,14 s

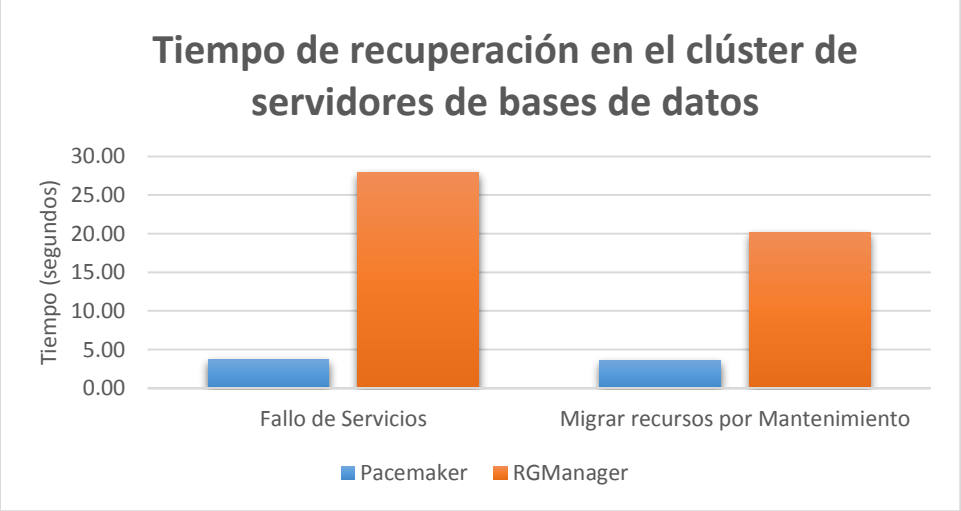


Figura 5.3. Tiempos de recuperación en el clúster de servidores de bases de datos

Tabla 12. Tiempos de recuperación en el clúster de servidores web

	CRM	
Técnicas de inyección de fallo	**Pacemaker**	**RGManager**
Fallo de Servicios	6,33 s	28,90 s
Fallo de Nodos	9,43 s	--
Migrar recursos por mantenimiento	2,60 s	20,42 s

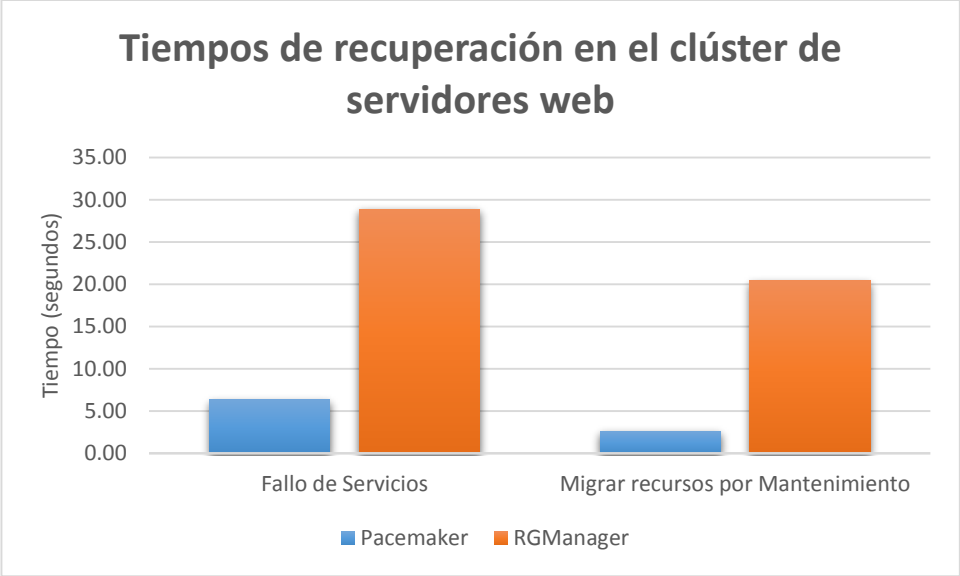


Figura 5.4. Tiempos de recuperación en el clúster de servidores web

Se puede observar que Pacemaker tiene tiempos de respuesta ante un fallo menores que los tiempos que tuvo RGManager. Esto se debió en parte al tiempo de monitoreo de 10 segundos que se configuró para cada recurso. Por esta razón se modificó el tiempo de monitoreo de recursos a 20 segundos y posteriormente a 60 segundos, como se muestra en la tabla 13, y se ilustra en la figura 5.5.

Tabla 13. Tiempo de recuperación vs tiempo de monitoreo en Pacemaker para la técnica de inyección de fallo a nivel de servicios

	Tiempo Recuperación	
Tiempo de monitoreo	**Clúster web**	**Clúster de bases de datos**
10 s	6.33 s	3.67 s
20 s	14.49 s	6.70 s
60 s	29.60 s	27.64 s

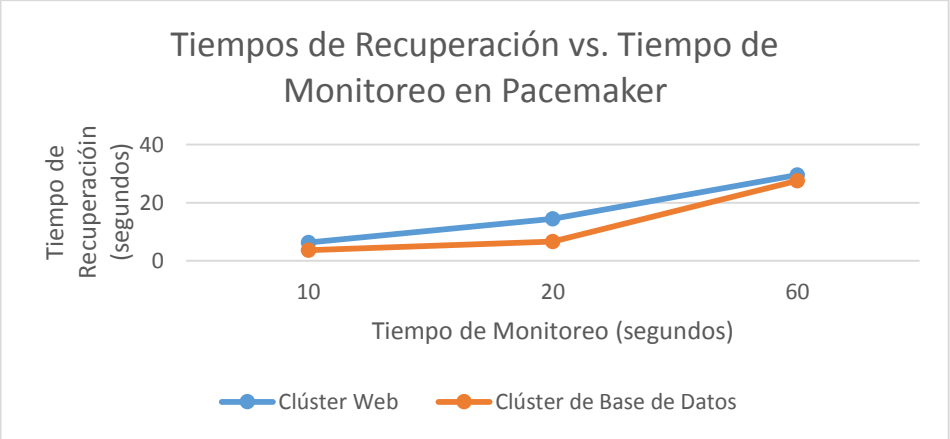


Figura 5.5. Tiempo de Recuperación vs Tiempo de Monitoreo en Pacemaker para la técnica de inyección de fallo a nivel de servicios

Como se observó en la Figura 5.5, el tiempo de recuperación aumenta en función del tiempo de monitoreo, lo cual permite concluir que el tiempo de respuesta es directamente proporcional al tiempo de monitoreo. A pesar de variar el tiempo de monitoreo hasta un valor de 60 segundos, Pacemaker siguió presentando un mejor tiempo de respuesta que RGManager con un

valor de 27,64 comparado con 27,94 segundos en el caso del clúster de bases de datos. Solo en el caso del clúster web, RGManager presentó un menor tiempo de respuesta, comparado a Pacemaker con valores de 28,90 segundos y 29,60 segundos, respectivamente.

5.2. Selección de la plataforma de virtualización y clúster para la solución de alta disponibilidad.

Luego de haber realizado las pruebas y el análisis de los resultados obtenidos de las mismas para las plataformas de virtualización y para las soluciones de clúster, en esta sección se describirán las razones por las cuales se descartaron ciertas soluciones y se eligió las que finalmente formaron parte de la solución de alta disponibilidad para los servidores de la FIEC. Con respecto a la plataforma de virtualización que formaría parte de la solución final, las pruebas de rendimiento jugaron un papel muy importante al momento de decidir cuál sería la plataforma de virtualización que se elegiría. El hipervisor Xen en el modo de virtualización completa, presentó un excesivo consumo de memoria RAM al momento de ejecutarse la prueba ab, además presentó un consumo total de CPU de 74,13% con la misma prueba, lo cual mostró que Xen en este modo consumía demasiados recursos, motivo por el cual fue descartado. Por otro lado, Xen en modo de paravirtualización presentó un menor consumo de memoria RAM en comparación con Hyper-V, en las dos pruebas de carga de trabajo. Sin embargo, no se dio así en el consumo de cpu, puesto que en la prueba ab, Hyper-V consumió un 61,75% en total comparado con Xen en modo de

paravirtualización con un 66,09%. En la prueba stress mientras que Xen en modo de paravirtualización presentó un menor consumo de CPU en el modo de ejecución del sistema, Hyper-V presentó un consumo menor de CPU en el modo de ejecución usuario, lo cual debió esperarse, ya que la prueba stress hacía que el CPU fuese consumido en un 100%, lo cual implicaba que la suma del consumo de CPU en los modos de ejecución del sistema y de usuario, deberían totalizar este porcentaje. De todo esto se puede concluir que Xen en modo de virtualización consume menor cantidad de memoria y Hyper-V consume menor cantidad de CPU, en situaciones de stress. Además de analizar el rendimiento de estas soluciones también se debió tomar en cuenta que el sistema operativo Windows Server 2012 R2, donde reside el rol Hyper-V requiere de una licencia que debe ser adquirida a cambio de un costo, a diferencia de Xen que es libre, y por lo tanto no requiere costo alguno para su adquisición. Otro factor importante es que al ser Xen de código abierto, éste está disponible para ser desarrollado por usuarios o programadores, para solucionar los problemas existentes o a su vez mejorar ciertas funcionalidades, lo cual es muy importante en instituciones educativas como la ESPOL, que fomentan la investigación de proyectos. Aunque Hyper-V es mucho más sencillo de usar debido a su interfaz gráfica, se pudo observar que también hay herramientas de libre distribución como libvirt, que permiten administrar el manejo de máquinas virtuales a través de una interfaz gráfica, lo cual facilita las tareas de administración de los servidores. Debido a todas estas

razones, se decidió que el hipervisor con el que se trabajaría para la solución final sería Xen en modo de paravirtualización.

Por su parte, los resultados obtenidos de las pruebas que se hicieron para las dos soluciones de clúster, mostraron que Pacemaker en todos los casos presentó un tiempo de respuesta menor ante un fallo en comparación con los tiempos de recuperación de RGManager, y solo en el caso en el que se varió el tiempo de monitoreo de los recursos para la técnica de inyección de fallo a 60 segundos, en el clúster de servidores Web, RGManager presentó un mejor tiempo de respuesta que Pacemaker. Como se explicó previamente, esto se debió a que Pacemaker realiza un monitoreo de los recursos de manera individual, tiempo de monitoreo que se puede incrementar o disminuir, mientras que el mecanismo que usa RGManager consiste en ir monitoreando cada recurso y servicio de su lista durante un tiempo de 10 segundos, lo cual implica que después de que un recurso pasa por su tiempo de monitoreo de 10 segundos deberá esperar que los otros recursos pasen su tiempo de monitoreo, para volver a ser revisado. En base a los resultados de los tiempos de respuesta se concluyó que la mejor opción para la solución de clúster que debía integrar la solución final de alta disponibilidad para los servidores de la FIEC, es la combinación de Pacemaker, CMAN y Corosync. Además que al momento de realizar las pruebas RGManager no realizó la conmutación de los servicios al momento de que se inyectó fallo al nodo, a pesar de luego haber configurado servicios muy básicos, con el objetivo de que la conmutación se diera. El hecho de que RGManager no permita realizar configuraciones de los

servicios de manera compleja, puesto que puede presentar comportamiento de conmutación inusual, fue otro de los motivos por el cual se la descartó como solución. Es así como luego de haber interactuado con las tres plataformas de virtualización y las dos soluciones de clúster, se concluyó que la plataforma de virtualización Xen en modo de Paravirtualización junto a la solución de alta disponibilidad de Pacemaker, CMAN y Corosync, trabajarían en conjunto para lograr la implementación de la solución de alta disponibilidad para los servidores de la FIEC.

5.3. Diseño de la Solución de Alta Disponibilidad.

Luego de haber seleccionado Xen en modo de Paravirtualización como la plataforma de virtualización y Pacemaker con CMAN y Corosync como la solución de clúster que integrarían la solución de alta disponibilidad final, a continuación se describirá el diseño de la misma. Se utilizó la plataforma de virtualización para instalar los servidores virtuales web y de bases de datos en cada uno de los servidores físicos, como se muestra en la Figura 5.6, donde se muestra el diagrama de la topología lógica del funcionamiento de la solución de alta disponibilidad. Los dos servidores virtuales de bases de datos instalados en los servidores físicos, uno en cada máquina, fueron los nodos que formaron el clúster de servidores de bases de datos. Este clúster tuvo una IP en común denominada IP Virtual, a través de la cual los servidores web tendrían acceso a la bases de datos, de manera que en el caso de fallo del nodo activo del clúster de los servidores de bases de datos, los servicios se migren al nodo pasivo, sin que los servidores web lo noten,

puesto que ellos simplemente seguirían apuntando sus peticiones a la dirección IP Virtual, como si de un solo servidor se tratara. Los otros dos servidores virtuales Web también fueron instalados, uno en cada servidor físico. Asimismo estos dos servidores virtuales fueron los nodos que formaron el clúster de servidores web, el cual también tuvo una IP virtual a través de la cual los clientes tuviesen acceso a los diferentes servicios web, que presta la facultad. De igual modo, esta IP ayudó a que cuando el cliente esté accediendo a los servicios web y haya una migración de recursos, el proceso sea completamente transparente para el usuario y no note la conmutación.

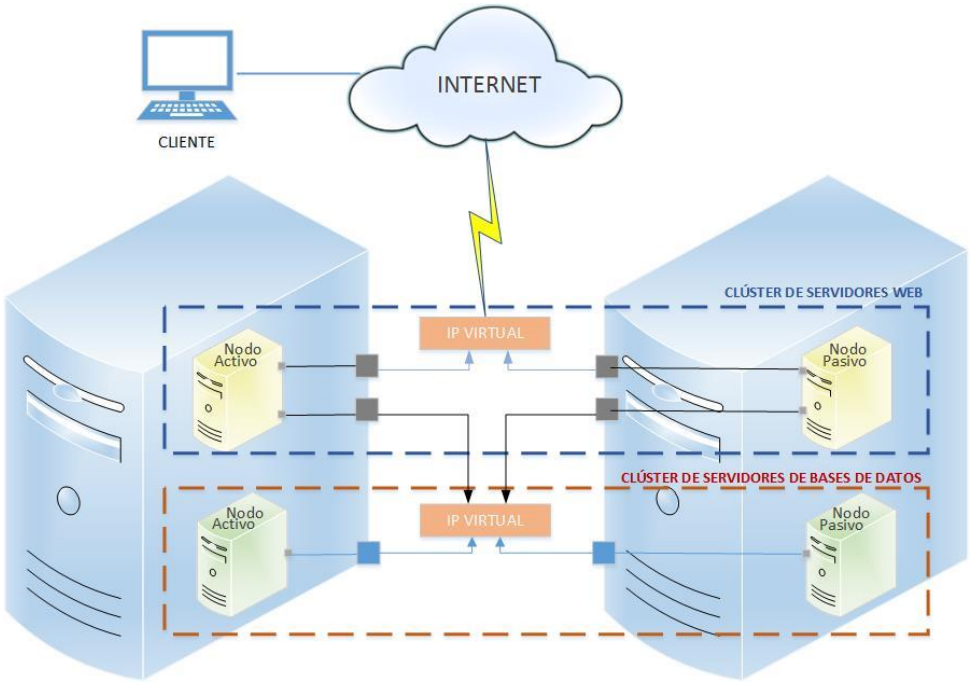


Figura 5.6. Diagrama de la topología lógica de la solución de alta disponibilidad para los servidores de la FIEC.

Otro de los factores que se definió al momento de crear el diseño para probar las diferentes soluciones de clúster, es la forma en la que los nodos tendrían acceso a la misma información. Esto es importante debido a que si el nodo activo llegaba a fallar, el nodo pasivo debía empezar a tomar el control de los servicios y para ello necesitaba acceder a la información actualizada ya sea correspondiente a las aplicaciones web que estaban siendo ejecutadas o a la base de datos que estaba siendo modificada por alguna transacción hecha por el usuario. Existen varias formas de lograr aquello, una de ellas es que los nodos del clúster trabajen con un software de replicación de datos como lo es *DRBD*, el cual permite, como su nombre lo indica, la replicación de los datos a través de la red [48]. Otra de las opciones es que compartan un mismo almacenamiento, y para ello se podía hacer uso del protocolo iSCSI, definiendo al dispositivo del almacenamiento compartido como el target o servidor iSCSI y a los nodos de los clústeres como los clientes iSCSI quienes tendrían acceso a dicho almacenamiento. Debido al retardo que ocasiona replicar los datos de un dispositivo a otro se decidió trabajar con el protocolo iSCSI. De esta forma todas las aplicaciones y bases de datos se guardarían en este almacenamiento y en el caso de fallo de alguno de los nodos de los clústeres, el otro nodo inmediatamente no solo reestablecería los servicios, sino que accedería a la información actualizada. Debido a las limitantes de hardware que se tuvo en la implementación de este proyecto, el equipo que albergó el almacenamiento compartido fue la computadora de escritorio con procesador Intel ® Core 2 Duo. En esta computadora se crearon asimismo

dos máquinas virtuales, cada una de ellas sería el servidor iSCSI que almacenaría la información para el clúster de servidores web y para el clúster de servidores de bases de datos, respectivamente. No se contempló la opción de realizar el backup de la información guardada en el almacenamiento compartido, lo cual suele servir para la recuperación de desastres, debido a que se encuentra fuera del objetivo principal de este proyecto. No obstante, es recomendable que se contemple la implementación del backup del almacenamiento compartido en futuras implementaciones.

De esta forma, cada uno de los nodos de los clústeres utilizó un iniciador iSCSI, que les permitió conectarse a los dispositivos SCSI (Target). Este dispositivo SCSI fue formateado con el sistema de archivos ext4. Hay otros sistemas de archivos como GFS2 u OCFS, que fueron creados para ser utilizados específicamente en clústeres. Sin embargo debido a que el clúster que se diseñó constaba solo de dos nodos, y tan solo uno a la vez accedería al almacenamiento compartido, se formateó el dispositivo SCSI con el sistema de archivos ext4.

La figura 5.7 muestra la topología física de la implementación de la solución de alta disponibilidad. En ella, se puede apreciar los tres computadores que fueron los equipos donde se alojaron los nodos del clúster y el almacenamiento compartido. Los servidores físicos, donde están instalados los nodos del clúster, poseían dos interfaces: una de ellas que permitió la comunicación con la red A y la otra que permitió la comunicación con la red

B. A través de La red A los nodos pudieron monitorearse continuamente con la técnica Heartbeat, y además a través de esta red tuvieron acceso al almacenamiento compartido. Por su parte las interfaces conectadas a la red B, permitieron que los equipos conectados a ellas pudiesen tener acceso a internet.

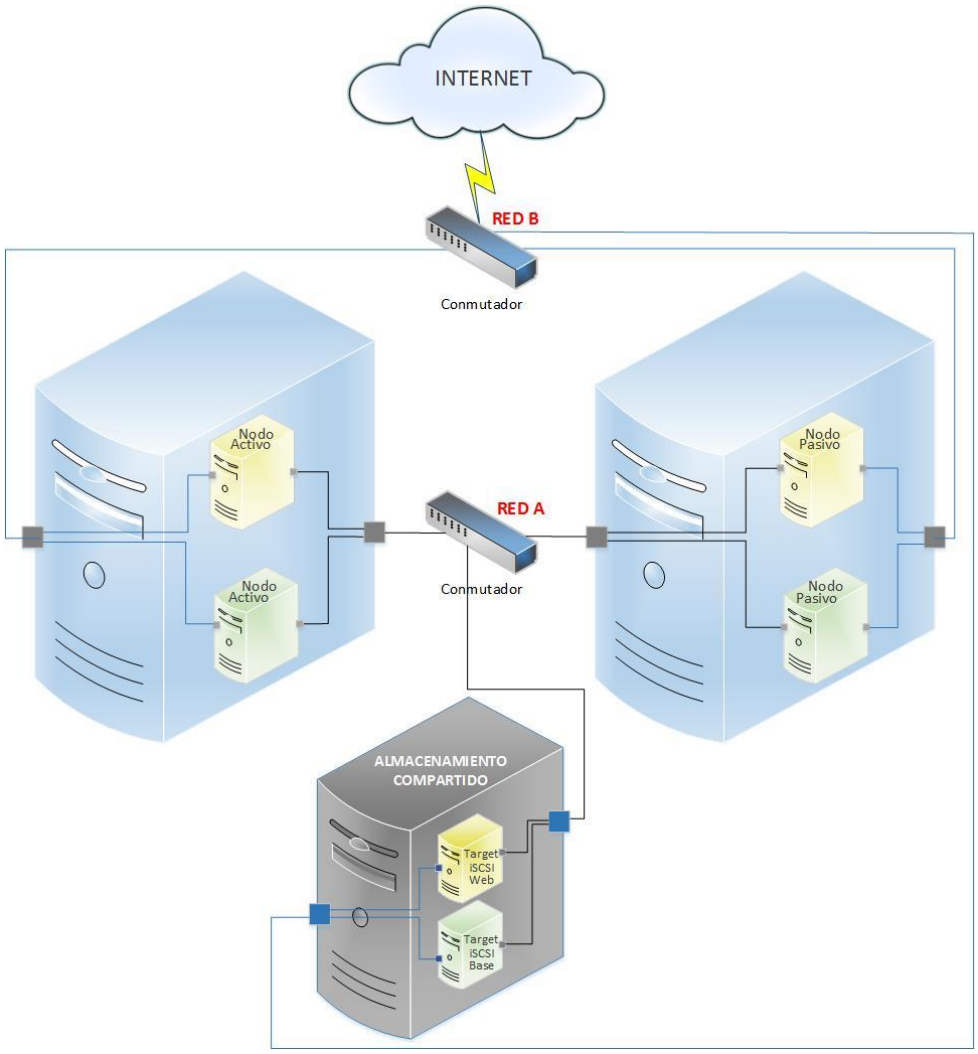


Figura 5.7. Diagrama de la topología física del diseño de alta disponibilidad

La instalación de los nodos del clúster en las máquinas virtuales fue similar a la que se realizó al momento de implementar Xen en modo de paravirtualización, con la diferencia de que las máquinas virtuales que alojarían los nodos del clúster contaron con 3Gb de Memoria RAM y dos CPUs virtuales cada una. Asimismo la instalación de la solución de clúster del stack conformado por Pacemaker, CMAN y Corosync, fue similar a la que se hizo en el momento de implementar las pruebas de las soluciones de clúster. A la configuración del clúster se le añadió los recursos iscsiUnit y iscsi-mount, los cuales se encargarían de establecer la conexión con el servidor iSCSI para acceder al almacenamiento compartido y de montar el dispositivo iSCSI en el directorio /mnt/storage previamente creado. También se establecieron las restricciones respectivas para que todos los recursos se ejecuten en el mismo nodo y en un orden específico, el cual fue muy importante definir puesto que no era posible que se ejecuten los recursos Tomcat, Httpd y iscsi-mount, si no se ejecutaba primero el recurso iscsiUnit. La configuración de las IPs para cada uno de los nodos del clúster se mantuvo igual a la que se realizó en la implementación de las pruebas de las soluciones de clúster. Finalmente, de todas las aplicaciones que se mencionaron en el capítulo 3, se procedió a instalar Nemesis, Reservar Salas, Certifiec, ARAFIEC y SATT. La instalación del resto de aplicaciones no fue posible, debido a restricciones de seguridad, según explicó el administrador de redes del Departamento de Soporte Técnico de la FIEC.

5.4. Implementación de la Solución.

Para la implementación de la solución final, se procedió a modificar la configuración de hardware virtual de los nodos que conformarían el clúster. Con la herramienta gráfica Virt-manager de libvirt, fue posible aumentar la capacidad de memoria RAM a 3Gb, y cambiar el número de CPUs virtuales de uno a dos. Toda esta configuración se la realizó con la plataforma de virtualización Xen en modo de paravirtualización, el cual se instaló sobre el Sistema Operativo Fedora 20. Debido a que previamente se hizo la instalación del stack Pacemaker, CMAN y Corosync para la implementación de las pruebas de las soluciones del clúster, descrita en el capítulo anterior, a toda esa instalación y configuración solo se adicionó la creación de los recursos iscsiUnit y iscsi-mount, que permitirían tener acceso al almacenamiento compartido, y el cambio de las restricciones de orden y de permanencia en los nodos para los recursos, como se indica en el anexo D, en la sección de configuraciones adicionales para la creación de la solución final de alta disponibilidad. Las configuraciones finales de los clústeres de servidores web y servidores de bases de datos, se muestran también en el Anexo D, en la sección Archivos de Configuración Final del Clúster. Finalmente se procedió a instalar las aplicaciones mencionadas en el diseño, las cuales son Nemesis, Reservar Salas, Certifiec, ARAFIEC y SATT, las cuales se alojaron en el almacenamiento compartido, junto con sus respectivas bases de datos.

CONCLUSIONES Y RECOMENDACIONES

Las conclusiones del presente proyecto son las siguientes:

1. Las plataformas de virtualización permiten un ahorro de los recursos de hardware que se usan dentro de una infraestructura tecnológica.

2. La plataforma de virtualización Xen en modo de paravirtualización, presentó un menor consumo de memoria RAM en promedio, que las otras dos plataformas al momento de realizar la prueba de carga de trabajo ab.

3. La tecnología de paravirtualización permite que el rendimiento de las máquinas virtuales sea mayor, debido a que no es necesario la simulación de todo el hardware, puesto que los drivers que éstas poseen permite que puedan trabajar con la mayoría del hardware del sistema anfitrión de manera directa, sin tener que requerir de la ayuda del hipervisor.

4. La plataforma de virtualización Hyper-V presentó un menor consumo de CPU en promedio, que las otras dos plataformas de virtualización al realizar la prueba de carga de trabajo stress.

5. La virtualización completa con hardware asistido, permite un mejor rendimiento de los sistemas invitados, debido a las extensiones introducidas en la arquitectura del procesador.

6. El stack conformado por Pacemaker, CMAN y Corosync, tuvo un menor tiempo de respuesta en promedio en todas las técnicas de fallo realizadas, comparado con el complemento de Alta Disponibilidad de RedHat.

7. El tiempo de recuperación de los servicios ante un fallo aumenta en función del aumento del tiempo de monitoreo de los servicios, con el stack conformado por Pacemaker, CMAN y Corosync.

8. El complemento de alta disponibilidad de RedHat, demora más tiempo en restaurar los servicios que sufrieron algún fallo, debido a que utiliza una lista con todos los recursos y servicios y los va monitoreando uno a uno durante un periodo de 10 segundos, de manera que luego de monitorear un recurso, éste debe esperar a que los otros sean monitoreados según el orden de la lista que posee RGManager.

9. Pacemaker realiza el monitoreo de los recursos de manera individual, de manera que si uno de los servicios falla, Pacemaker lo detectará de forma más rápida y por consiguiente el tiempo de respuesta ante un fallo será menor que el de RGManager.

10. El orden en el que se ejecutan los recursos dentro de un clúster es muy importante, debido a la dependencia que muchos de ellos tienen entre sí.

11. La ejecución de todos los recursos dentro del mismo nodo del clúster, es importante, debido a la dependencia que muchos de ellos tienen entre sí.

12. A pesar de que la administración con la interfaz gráfica que ofrece Conga es más sencilla, se requiere de otras interfaces por líneas de comando para realizar tareas de monitoreo y de detección de problemas existentes en el clúster.

13. El desarrollo de nuevas tecnologías con software libre, permiten la implementación económica y eficaz de soluciones que permiten aumentar el rendimiento y la disponibilidad de los servidores.

Las recomendaciones del presente proyecto son las siguientes:

1. Realizar en un futuro la implementación de este proyecto sobre servidores físicos especializados, que permitan agregar funcionalidades que por limitaciones de los equipos usados, no se pudo realizar.

2. Utilizar hardware más potente que permita una rápida comunicación entre los nodos que conforman el clúster.

3. Crear configuraciones de redes redundantes que permitan fortalecer la solución de alta disponibilidad.

4. Agregar la implementación de una solución de recuperación de desastres, que permita realizar un backup del almacenamiento compartido y así fortalecer la implementación de la solución de alta disponibilidad.

GLOSARIO

RAID. Sistema de almacenamiento de datos, que distribuye o replica los datos que se encuentren en las unidades de almacenamiento de datos.

Gestor de Volúmenes Lógicos. Método que permite virtualizar los discos físicos de manera que se pueda tener varios discos lógicos con la posibilidad de redimensionarlos.

SAN. Arquitectura de almacenamiento que necesita redes de alta velocidad dedicadas solo para almacenamiento y Backup, optimizada para mover grandes cantidades de datos, y consistente en múltiples recursos de almacenamiento geográficamente distribuidos. Suelen basarse en la tecnología FC, aunque también puede basarse en tecnología Gigabit Ethernet con iSCSI.

Canal de Fibra. Tecnología de red utilizada principalmente para redes de almacenamiento. La señalización del canal puede funcionar sobre pares de cobre o cables de fibra óptica.

Internet iSCSI. Estándar que permite el uso del protocolo SCSI sobre redes TCP/IP.

SCSI. Interfaz estándar para la transferencia de datos entre distintos dispositivos del bus de la computadora.

Computación grid. Tecnología que permite usar de forma coordinada todo tipo de recursos, que no están sujetos a un control centralizado.

Computación en la nube. Sistema informático basado en Internet y Centros de datos remotos para gestionar servicios de información y aplicaciones.

Hipervisor o Monitor de máquina virtual. Es una plataforma que emplea técnicas de control de virtualización.

Anillo de protección. Mecanismo para proteger datos y funcionalidad de los fallos. Están dispuestos en una jerarquía de la más privilegiada a la menos privilegiada, siendo el anillo 0 el que posee mayores privilegios.

Infraestructura de escritorio virtual. Es la práctica de hospedar un sistema operativo para computadoras de escritorio en una máquina virtual.

Chroot. Operación que cambia el directorio raíz aparente para el proceso de ejecución actual y sus hijos.

Protocolo RDP. Protocolo propietario desarrollado por Microsoft que permite la comunicación en la ejecución de una aplicación entre un terminal y un servidor Windows.

Protocolo X Window. Protocolo que permite la comunicación en la ejecución de una aplicación entre un terminal y un servidor Linux.

Nodo. Es cada una de las máquinas que se encuentran interconectadas dentro de la red que forma el clúster, de iguales o diferentes características.

Sistema Operativo. Es un entorno multiusuario y multiproceso, de tipo servidor, con capacidad de abstracción de dispositivos y que trabaja con interfaces IP virtuales.

Middleware. Es el software que actúa entre el sistema operativo y las aplicaciones, brindando al usuario la experiencia de estar utilizando una única súper máquina. Optimiza el sistema y provee herramientas de mantenimiento para procesos pesados como migraciones, balanceo de carga, tolerancia de fallos, etc.

Sistema de Almacenamiento. Sistema que permite almacenar y procesar información.

Heartbeat. Técnica que permite conocer en todo momento la disponibilidad de los equipos físicos, a través del informe periódico de su existencia al resto de nodos mediante una “señal de vida”.

Fencing. Es la técnica que le permite al clúster conocer que un nodo está funcionando de manera incorrecta, de manera que el clúster proceda a retirarle los recursos asignados para que sean atendidos por los otros nodos del clúster y lo deja en estado inactivo, para evitar que el nodo corrompa recursos o responda con peticiones.

Split-Brain. Escenario que ocurre cuando más de un servidor o aplicación del mismo cree que los demás servidores han fallado e intenta activar y utilizar los recursos que se están ejecutando en el servidor que considera en estado fallido, lo que puede ocasionar un daño a tales recursos.

Quorum. Mecanismo que se encarga de decidir que partición del clúster continuará funcionando a través de un algoritmo que se basa en los “votos” de los nodos. Se da a cada nodo un voto y aquella partición que cuente con la mayoría de votos tendrá quórum y continuará funcionando.

BackendDriver. Interfaz que hace peticiones para acceder al hardware, permitiendo así que varios domUs puedan compartir tal hardware. Se aloja en el dom0.

FrontendDriver. Interfaz alojada en el domU que usa XenBus, XenStore, paginas compartidas, notificaciones de eventos para comunicarse con el BackendDriver, quien se encarga de cumplir dichas peticiones.

XenBus. Driver que proporciona una abstracción para los conductores de autobuses-virtualizados para comunicarse entre dominios.

XenStore. Espacio de almacenamiento de información compartida entre los dominios mantenidos por el Xenstored.

Modo invitado. Modo de ejecución normal para el código del sistema invitado siempre que no tenga operaciones de entrada/salida.

Modo usuario. Modo de ejecución usado para ejecutar las operaciones de entrada/salida del sistema invitado y permitirá gestionar dispositivos virtuales a nivel de usuario.

Modo núcleo. Modo de ejecución que se usa para trabajar en modo invitado y para gestionar las salidas desde modo usuario, causadas por operaciones especiales o de entrada/salida.

Live Migration. La migración en vivo es el proceso de mover o ejecutar máquinas virtuales o aplicaciones entre diferentes máquinas físicas sin desconectar al cliente o a las aplicaciones.

Quick Migration. Es una característica disponible en Windows Server 2008 que permite mover máquinas virtuales entre nodos con el mínimo tiempo de inactividad.

Storage Migration. Es el proceso de mover el almacenamiento de una máquina virtual sin tiempo de inactividad.

Active Directory. Es la implementación de servicio de directorio en una red distribuida de computadores que usa Microsoft.

STONITH. Es una técnica que aísla un nodo fallido en el clúster. Para lograr aquello procede a resetearlo o a apagarlo.

DLM. Es aquel que provee aplicaciones de software distribuido con un medio para sincronizar el acceso a los recursos compartidos.

GFS. Sistema de archivos distribuidos para clústeres de servidores que permite que todos los nodos tengan acceso simultáneo directo al mismo bloque de almacenamiento compartido.

CLVM. Es un conjunto de extensiones de clúster para LVM. Estas extensiones permiten al clúster administrar almacenamiento compartido usando LVM.

Add-on. Es una pieza de software que mejora otra aplicación de software y por lo general no se puede ejecutar de manera independiente.

Plugin. Es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.

XML. Es un lenguaje de marcas utilizado para almacenar datos en forma legible. Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

Crmsh. Es una interfaz de línea de comando para clúster de alta disponibilidad que permite configurar, administrar y solucionar problemas del clúster.

PC. Interfaz de línea de comandos basada en Web, que permite manejar todo el clúster, desde su instalación hasta la configuración y el estado de los recursos.

Pygui. Interfaz gráfica de usuario para Pacemaker escrito en Python por IBM China.

Halcón. Interfaz gráfica de usuario basada en la web para la gestión y seguimiento de clústeres de alta disponibilidad de Pacemaker.

LCMC. Interfaz gráfica de usuario que representa las situaciones y relaciones entre los servicios del clúster. Utiliza SSH para instalar, configurar y administrar clústeres.

Libvirt. Es una *API,* que permite administrar varias soluciones de Virtualización como KVM y Xen a través de una Interfaz (de programación y/o usuario) común.

API. Conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software.

Virt-manager. Administrador gráfico de máquinas virtuales que permite crear/administrar máquinas virtuales, las redes a las éstas estarán conectadas y el almacenamiento que utilizarán.

Puente de Red. Dispositivo de interconexión de redes de computadores que opera en la capa 2 (nivel enlace de datos) del modelo OSI.

NTPD. Es un programa del sistema operativo que establece y mantiene la hora del sistema en sincronización con servidores de tiempo estándar en Internet.

SELinux. Es un módulo de seguridad para el kernel de Linux que proporciona el mecanismo para soportar políticas de seguridad para el control de acceso.

Conga. Es una arquitectura agente/servidor para la administración remota de los sistemas.

Ricci. Es el agente de la arquitectura agente/servidor Conga.

Luci. Es el servidor de la arquitectura agente/servidor Conga, que puede comunicarse con múltiples agentes Ricci en los sistemas y al cual se accede a través de un navegador usando https.

Log. Es un registro de la actividad de un sistema, que generalmente se guarda en un fichero de texto.

NetworkManager. Programa que proporciona a los sistemas la detección y configuración automática para conectarse a la red.

DRBD. Es una solución de almacenamiento replicada, que no comparte, basada en software que refleja el contenido de dispositivos de bloque (discos duros, particiones, volúmenes lógicos, etc) entre servidores.

ANEXOS

Anexo A: Estudio preliminar

Para definir el número de repeticiones de las pruebas se utilizó la fórmula A-1, denominada fórmula del tamaño de muestra, donde $\boldsymbol{n}$ es el tamaño de la muestra, $\boldsymbol{Z_{\alpha/2}}$ es el porcentaje de confiabilidad, $\boldsymbol{\sigma}$ es la desviación estándar de la población y $\boldsymbol{e}$ es el error estimado.

$$n = \frac{Z_{\alpha/2}{}^2 \times \sigma^2}{e^2} \qquad \text{(A.1)}$$

Para las pruebas se tomó como constantes los siguientes parámetros: $\boldsymbol{Z_{\alpha/2} = 1,96}$ tomado en relación al 95% de nivel de confiabilidad, $\boldsymbol{\sigma = 0,5}$ el cual suele usarse cuando no se tiene especificado su valor y finalmente $\boldsymbol{e}$= 0,1 como error estimado para el método de carga de estrés de uso de recursos y para el método de carga de estrés de tráfico utilizando solicitudes web.

El resultado obtenido de la ecuación para el número de repeticiones de las pruebas es de $\boldsymbol{n}$ = 96 número de veces a probar.

Anexo B: Figuras

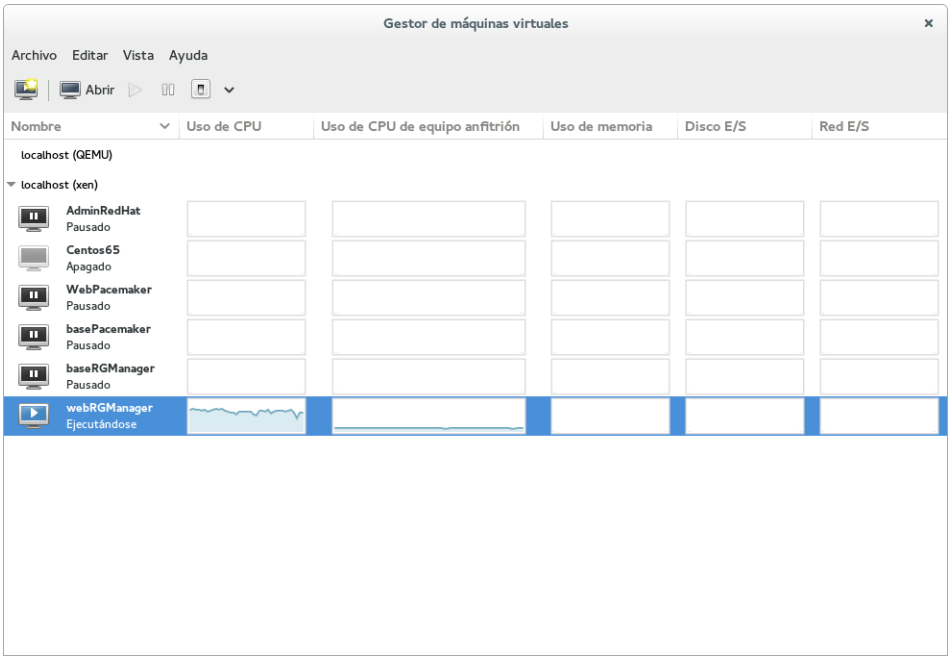


Figura B-1. Gestor de máquinas virtuales con Virt-manager

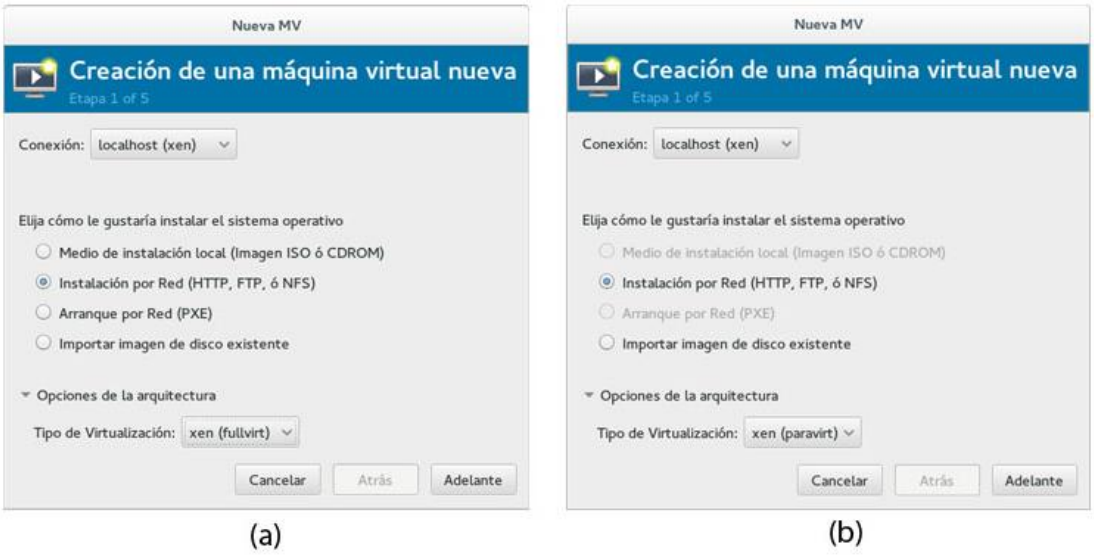


Figura B-2. Creación de máquinas virtuales con Xen (a) Modo de Virtualización Completa y (b) Modo de Paravirtualización

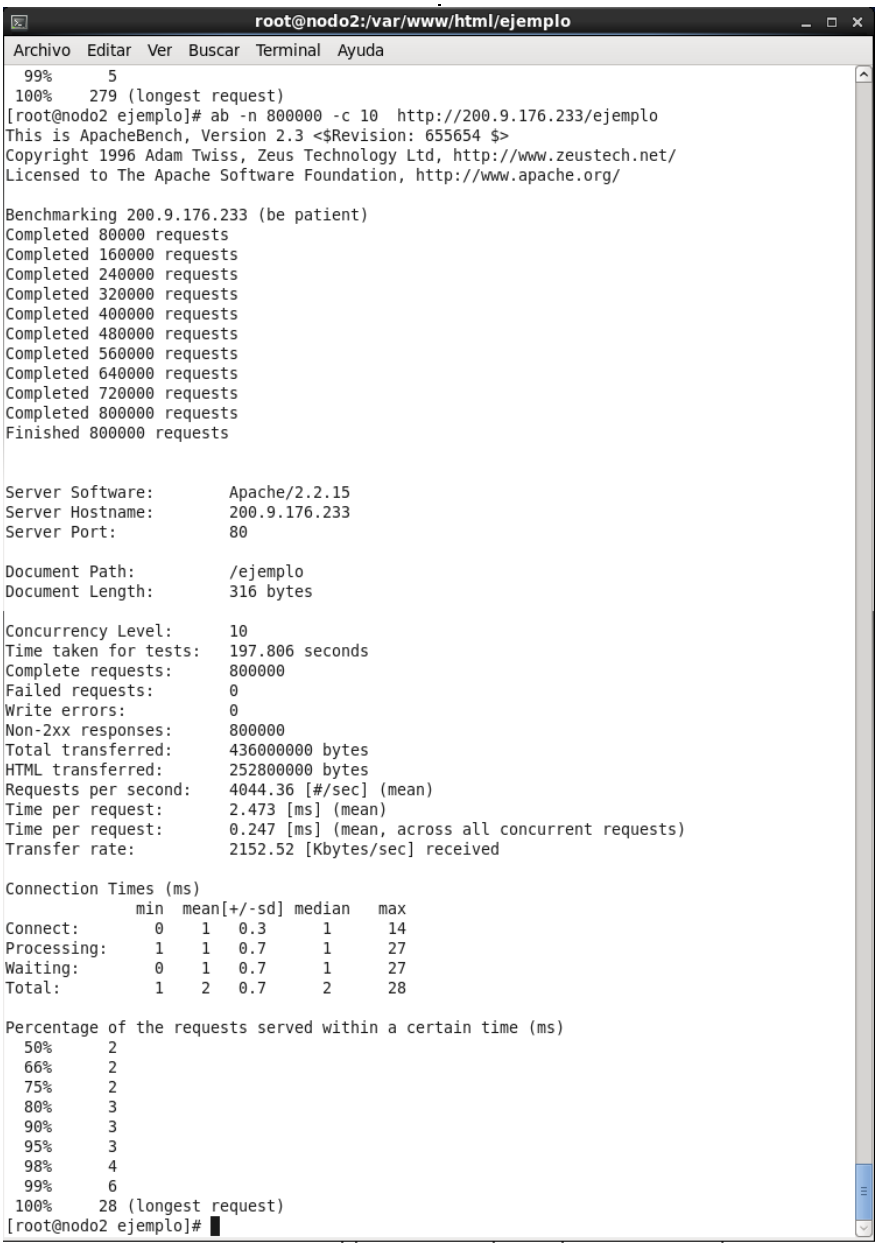


Figura B-3. Ejecución del comando ab

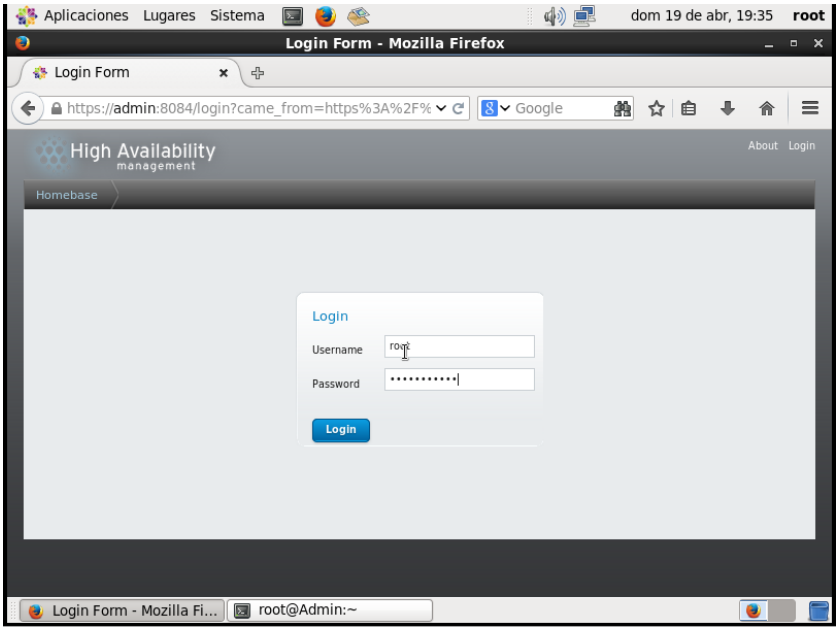


Figura B-4. Pantalla de ingreso a la Interfaz de Administración del clúster creado con el complemento de Alta Disponibilidad de RedHat

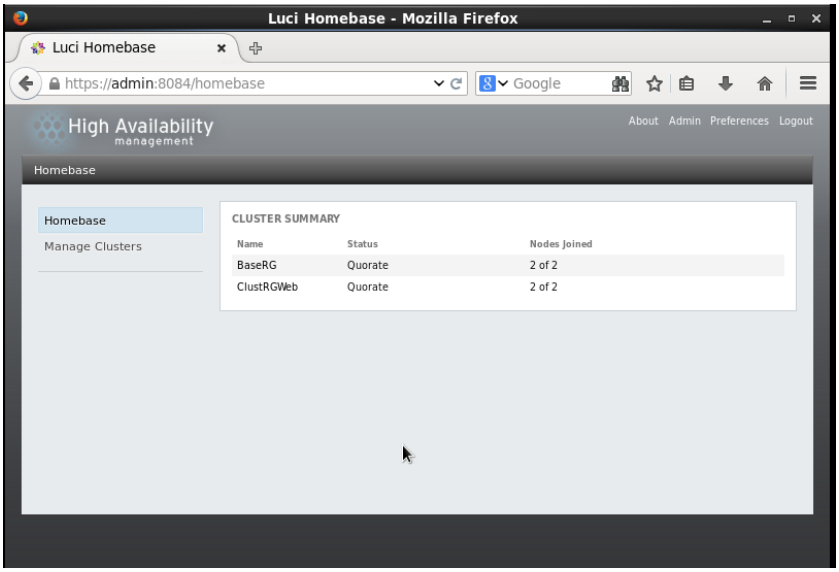


Figura B-5. Resumen de clúster creados con el complemento de Alta Disponibilidad de RedHat

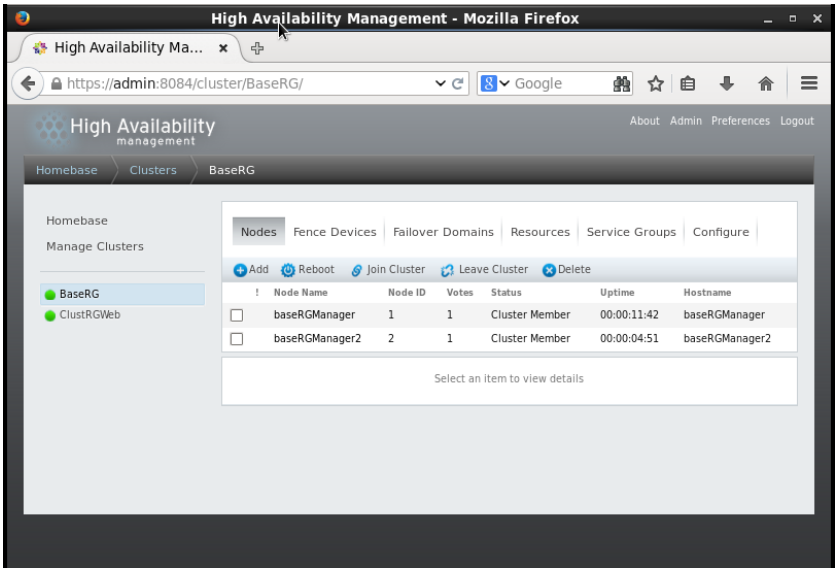


Figura B-6. Nodos del clúster creado con el complemento de Alta Disponibilidad de RedHat

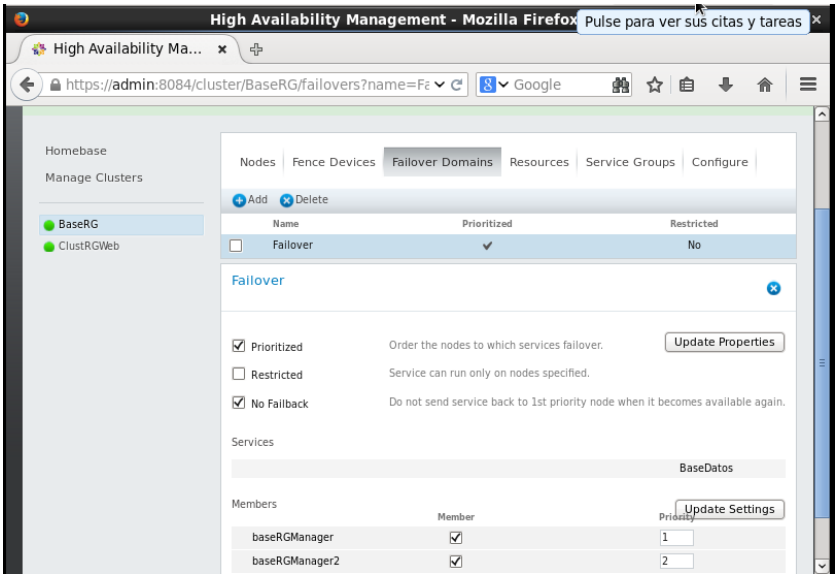


Figura B-7. Dominio de conmutación del clúster creado con el complemento de Alta Disponibilidad de RedHat

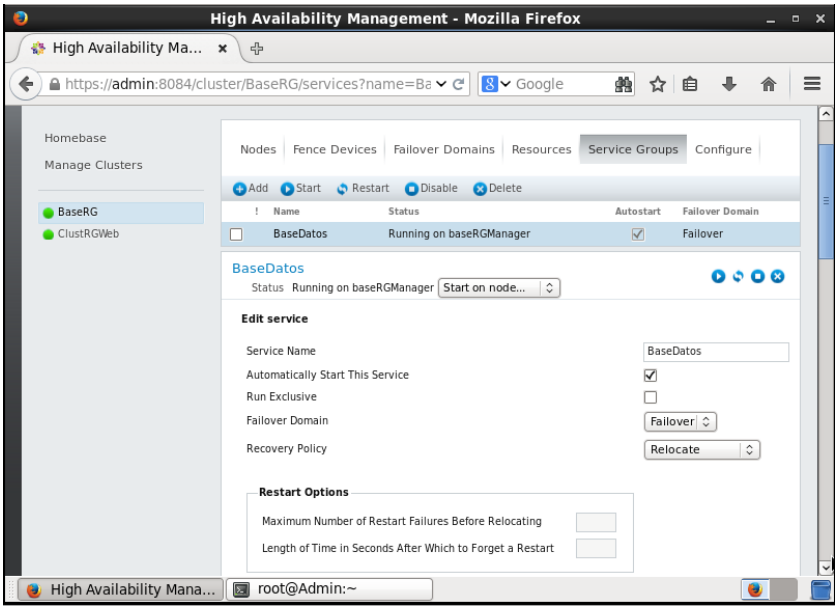


Figura B-8. Grupo de Servicios del clúster creado con el complemento de Alta Disponibilidad de RedHat

Anexo C: Herramientas de carga de trabajo y monitoreo

Stress. Es una herramienta que permite imponer carga de trabajo al procesador, memoria, operaciones de entrada/salida o al disco, durante un tiempo determinado. Los parámetros que el comando stress permite usar al momento de su ejecución se observa en la siguiente tabla.

Tabla 14. Parámetros del comando stress

Parámetros	**Definición**
--cpu [n]	Divide [n] número de procesos. Cada proceso calcula la raíz cuadrada de un número aleatorio en un lazo que se detiene al finalizar el tiempo definido por timeout. Es una manera de tener la CPU constantemente ocupada.
--io [n]	Divide [n] número de procesos que llama a la función sync en un lazo. Sync, vuelca los datos almacenados temporalmente en la memoria del disco. La ejecución de este procedimiento dará una idea del rendimiento de E/S.
--vm [n]	Bifurca [n] procesos para asignar y liberar memoria, en un lazo con las funciones malloc( ) y free( ).
--vm-bytes [n]	Permite controlar la carga en la memoria definiendo el tamaño de la misma.
--hdd [n]	Llama a la función *escritura* en un lazo.
--timeout [n]	Tiempo que dura el proceso de stress.

ab. Es una herramienta para la evaluación comparativa del servidor Apache. Para que ab funcione se le debe especificar la url destino a revisar, el número de conexiones y cuantas de ellas se realizarán simultáneamente. A continuación se detallarán los parámetros que se usaron en este proyecto para la ejecución de la herramienta ab.

Tabla 15. Parámetros del comando stress

Parámetros	Definición
-n [n]	[n] número de conexiones.
-c [n]	[n] conexiones concurrentes
http://direccion.com/	Dirección del servidor, que es objeto de la prueba de benchmarking

sar. Este comando muestra las estadísticas de la utilización de recursos como cpu o memoria, durante un tiempo determinado y en intervalos específicos. Si lo que se requiere es monitorear el uso de cpu, se escribirá el siguiente comando: **sar 2 150,** donde **2** es el intervalo de tiempo en segundos en el que mostrará estadísticas de uso del cpu y **150** es el número de estadísticas que se desea mostrar. Si por otro lado, se desea monitorear el uso de memoria ram, solo será necesario agregar el parámetro ***–r*** luego del comando sar, por ejemplo: **sar –r 2 150.**

Anexo D: Procesos de instalación y archivos de configuración final

Los valores [ALL] y [ONE] en cada una de las configuraciones, indican si tal configuración debe hacerse en cada uno de los nodos o solo en uno de ellos, respectivamente.

1. Configuración previa a la instalación de las soluciones de clúster

1.1. Deshabilitar NetworkManager

[ALL] # service NetworkManager stop

[ALL] # chkconfig NetworkManager off

1.2. Configurar módulo de seguridad SELinux

[ALL] # nano /etc/sysconfig/selinux

Cambiar el valor de SELINUX:

SELINUX=disabled

1.3. Agregar nombre de nodos a fichero /etc/hosts

[ip_Nodo1] [NombreNodo1]

[ip_Nodo2] [NombreNodo2]

1.4. Deshabilitar IpTables

[ALL] # service iptables stop

[ALL] # chkconfig iptables off

2. Creación de Clúster con Pacemaker, CMAN y Corosync

2.1. Instalación de paquetes del Clúster

[ALL] # yum install corosync pacemaker cman pcs ccs resource-agents

2.2. Instalación de interfaz de línea de comando Shell crmsh

[ALL] # yum -y install python-dateutil redhat-rpm-config

[ALL] # rpm -ivh http://download.opensuse.org/repositories/network:/ha-clustering:/Stable/CentOS_CentOS-6/x86_64/python-pssh-2.3.1-4.1.x86_64.rpm

[ALL] # rpm -ivh http://download.opensuse.org/repositories/network:/ha-clustering:/Stable/CentOS_CentOS-6/x86_64/pssh-2.3.1-4.1.x86_64.rpm

[ALL] # rpm -ivh http://download.opensuse.org/repositories/network:/ha-clustering:/Stable/CentOS_CentOS-6/x86_64/crmsh-2.1-1.6.x86_64.rpm

2.3. Creación del fichero cluster.conf

[ONE] # ccs -f /etc/cluster/cluster.conf --createcluster <NombreCluster>

[ONE] # ccs -f /etc/cluster/cluster.conf --addnode <NombreNodo1>

[ONE] # ccs -f /etc/cluster/cluster.conf --addnode <NombreNodo2>

[ONE] # ccs -f /etc/cluster/cluster.conf --addfencedev pcmk agent=fence_pcmk

[ONE] # ccs -f /etc/cluster/cluster.conf --addmethod pcmk-redirect <NombreNodo1>

[ONE] # ccs -f /etc/cluster/cluster.conf --addmethod pcmk-redirect <NombreNodo2>

[ONE] # ccs -f /etc/cluster/cluster.conf --addfenceinst pcmk

<NombreNodo1> pcmk-redirect port= <NombreNodo1>

[ONE] # ccs -f /etc/cluster/cluster.conf --addfenceinst pcmk

<NombreNodo2> pcmk-redirect port= <NombreNodo2>

2.4. Configuración de fichero CMAN

[ALL] # echo "CMAN_QUORUM_TIMEOUT=0" >> /etc/sysconfig/cman

2.5. Iniciar servicios cman, pacemaker y pcsd

[ALL] # service cman start

[ALL] # service pacemaker start

[ALL] # service pcsd start

[ALL] # chkconfig cman on

[ALL] # chkconfig pacemaker on

[ALL] # chkconfig pcsd on

2.6. Configuración de pcsd

[ONE] # passwd hacluster

[ONE] # pcs cluster auth <NombreNodo1> <NombreNodo2>

2.7. Configuraciones Adicionales

[ONE] # pcs property set stonith-enabled=false

[ONE] # pcs property set no-quorum-policy=ignore

[ONE] # crm configure rsc_defaults resource-stickiness=”INFINITY”

2.8. Creación de recursos con Pacemaker para el clúster de Base de Datos

- Creación de IP Virtual.

  [ONE] # pcs resource create VirtualIP ocf:heartbeat:IPaddr2

  ip="192.168.5.25" cidr_netmask=24

  [ONE] # pcs resource op add VirtualIP monitor interval=20s

- Creación de recurso mysql

  [ONE] # crm configure

  [ONE] # primitive BaseDatos lsb:mysqld \

  [ONE] # pcs resource op add BaseDatos monitor interval="10s"

  timeout="20s"

- Creación de restricciones para el cluster

  pcs constraint colocation add VirtualIP BaseDatos INFINITY

  pcs constraint order VirtualIP then BaseDatos

2.9. Creación de recursos con Pacemaker para el Clúster Web

- Creación de IP Virtual.

  [ONE] # pcs resource create VirtualIP ocf:heartbeat:IPaddr2

  ip="200.9.176.235" cidr_netmask=25

  [ONE] # pcs resource add_operation VirtualIP monitor interval=10s

- Creación de recurso Httpd

  [ONE] # pcs resource create Httpd ocf:heartbeat:apache params

  configfile="/etc/httpd/conf/httpd.conf" port="80"

  [ONE] # pcs resource op add Httpd start interval="0"

  [ONE] # pcs resource op add Httpd stop interval="0"

  [ONE] # pcs resource op add Httpd monitor interval="10s"

  timeout="20s"

- Creación de recurso Tomcat6

  [ONE] # crm configure primitive Tomcat6 ocf:heartbeat:tomcat

  crm (live) configure # params java_home=/usr/lib/jvm/jre

  crm (live) configure # catalina_home=/usr/local/tomcat6

  crm (live) configure # catalina_base=/usr/local/tomcat6

  crm (live) configure # op monitor interval="10s"

  crm (live) configure # op start timeout="160s"

  crm (live) configure # op stop timeout="160s"

3. Creación de Clúster con el complemento de alta disponibilidad de RedHat

3.1. Instalación de paquetes del Clúster

- En los nodos del clúster

  yum groupinstall "High Availability"

  yum install ricci

- En la máquina de administración del clúster

  yum groupinstall "High Availability Management"

  yum install ricci

3.2. Fichero cluster.conf inicial

```
<?xml version="1.0"?>
<cluster config_version="1" name=”NombreCluster">
    <clusternodes>
        <clusternode name="cnode1" nodeid="1"/>
        <clusternode name="cnode2" nodeid="2"/>
    </clusternodes>
</cluster>
```

3.3. Configuración de servicios en los nodos del clúster

```
chkconfig iptables off
chkconfig ip6tables off
chkconfig ricci on
chkconfig cman on
chkconfig rgmanager on
chkconfig modclusterd on
passwd ricci
service iptables stop
service ip6tables stop
service ricci start
```

service cman start

service rgmanager start

service modclusterd start

3.4. Configuración de servicios en la máquina de administración del clúster

chkconfig iptables off

chkconfig ip6tables off

chkconfig luci on

chkconfig ricci on

service iptables stop

service ip6tables stop

service luci start

service ricci start

4. Configuraciones adicionales para la creación de solución final de alta disponibilidad.

4.1. Eliminación de las restricciones hechas para la solución del stack Pacemaker, Corosync y CMAN.

pcs constraint order remove <idRestricción>

pcs constraint colocation remove <idRestricción>

4.2. Adición de los recursos iscsiUnit y iscsi-mount para acceder al almacenamiento compartido.

- Recurso iscsiUnit para clúster de Servidores Web

[ONE] # crm configure

[ONE] # primitive iscsiUnit lsb:iscsi params portal=”192.168.5.111”

target=”iqn.2015-03.com.cluster:tgt1”

op start interval=”0” timeout=”20”

op stop interval=”0” timeout=”20”

op monitor interval=”20” timeout=”30” start-delay=”0”

- Recurso iscsiUnit para clúster de Servidores de bases de datos

[ONE] # crm configure

[ONE] # primitive iscsiUnit lsb:iscsi \

params portal=”192.168.5.110”

target=”iqn.2015-02.com.cluster:tgt1”

op start interval=”0” timeout=”20”

op stop interval=”0” timeout=”20”

op monitor interval=”20” timeout=”30” start-delay=”0”

- Recurso iscsi-mount

[ONE] # crm configure

[ONE] # primitive iscsi-mount ocf:heartbeat:Filesystem

params device="/dev/sda1" directory="/mnt/storage" fstype="ext4"

op start interval="0" timeout="120" op stop interval="0" timeout="120"

4.3. Adición de restricciones al clúster

- Para el clúster de Servidores Web

```
pcs constraint colocation add VirtualIP iscsiUnit INFINITY
pcs constraint colocation add iscsiUnit iscsi-mount INFINITY
pcs constraint colocation add iscsi-mount Tomcat6 INFINITY
pcs constraint colocation add Tomcat6 Httpd INFINITY
pcs constraint order VirtualIP then iscsiUnit
pcs constraint order iscsiUnit then iscsi-mount
pcs constraint order iscsi-mount then Tomcat6
pcs constraint order Tomcat6 then Httpd
```

- Para el clúster de Servidores de Bases de Datos

```
pcs constraint colocation add VirtualIP iscsiUnit INFINITY
pcs constraint colocation add iscsiUnit iscsi-mount INFINITY
pcs constraint colocation add iscsi-mount BaseDatos INFINITY
pcs constraint order VirtualIP then iscsiUnit
pcs constraint order iscsiUnit then iscsi-mount
pcs constraint order iscsi-mount then BaseDatos
```

5. Archivos de Configuración Final del Clúster.

5.1. Clúster de Servidores Web

Cluster Name: WebPace

Corosync Nodes:

webPacemaker webPacemaker2

Pacemaker Nodes:

webPacemaker webPacemaker2

Resources:

Resource: VirtualIP (class=ocf provider=heartbeat type=IPaddr2)

Attributes: ip=200.9.176.235 cidr_netmask=25

Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)

stop interval=0s timeout=20s (VirtualIP-stop-timeout-20s)

monitor interval=10s timeout=20s (VirtualIP-monitor-interval-10s)

monitor interval=20s (VirtualIP-name-monitor-interval-20s)

Resource: iscsiUnit (class=lsb type=iscsi)

Attributes: portal=192.168.5.111 target=iqn.2015-03.com.cluster:tgt1

Operations: start interval=0 timeout=20 (iscsiUnit-start-0)

stop interval=0 timeout=20 (iscsiUnit-stop-0)

monitor start-delay=0 interval=20 timeout=30 (iscsiUnit-monitor-20)

Resource: iscsi-mount (class=ocf provider=heartbeat type=Filesystem)

Attributes: device=/dev/sda1 directory=/mnt/storage fstype=ext4

Operations: start interval=0 timeout=120 (iscsi-mount-start-0)

stop interval=0 timeout=120 (iscsi-mount-stop-0)

monitor interval=20s timeout=20s (iscsi-mount-name-monitor-interval-20s-timeout-20s)

Resource: Tomcat6 (class=ocf provider=heartbeat type=tomcat)

Attributes: java_home=/usr/lib/jvm/jre catalina_home=/usr/local/tomcat6 catalina_base=/usr/local/tomcat6

Operations: monitor interval=20s (Tomcat6-monitor-20s)

start interval=0 timeout=160s (Tomcat6-start-0)

stop interval=0 timeout=160s (Tomcat6-stop-0)

Resource: Httpd (class=ocf provider=heartbeat type=apache)

Attributes: configfile=/etc/httpd/conf/httpd.conf port=80

Operations: start interval=0s timeout=40s (Httpd-start-timeout-40s)

stop interval=0s timeout=60s (Httpd-stop-timeout-60s)

monitor interval=10 timeout=20s (Httpd-monitor-interval-10)

start interval=0 (Httpd-name-start-interval-0)

stop interval=0 (Httpd-name-stop-interval-0)

monitor interval=20s timeout=20s (Httpd-name-monitor-interval-20s-timeout-20s)

Stonith Devices:

Fencing Levels:

Location Constraints:

Ordering Constraints:

start VirtualIP then start iscsiUnit (kind:Mandatory) (id:order-VirtualIP-iscsiUnit-mandatory)

start iscsiUnit then start iscsi-mount (kind:Mandatory) (id:order-iscsiUnit-iscsi-mount-mandatory)

start iscsi-mount then start Tomcat6 (kind:Mandatory) (id:order-iscsi-mount-Tomcat6-mandatory)

start Tomcat6 then start Httpd (kind:Mandatory) (id:order-Tomcat6-Httpd-mandatory)

Colocation Constraints:

VirtualIP with iscsiUnit (score:INFINITY) (id:colocation-VirtualIP-iscsiUnit-INFINITY)

iscsiUnit with iscsi-mount (score:INFINITY) (id:colocation-iscsiUnit-iscsi-mount-INFINITY)

iscsi-mount with Tomcat6 (score:INFINITY) (id:colocation-iscsi-mount-Tomcat6-INFINITY)

Tomcat6 with Httpd (score:INFINITY) (id:colocation-Tomcat6-Httpd-INFINITY)

Cluster Properties:

cluster-infrastructure: cman

dc-version: 1.1.11-97629de

no-quorum-policy: ignore

stonith-enabled: false

5.2. Clúster de Servidores de Bases de Datos

Cluster Name: BasePace

Corosync Nodes:

BasePacemaker basePacemaker2

Pacemaker Nodes:

BasePacemaker basePacemaker2

Resources:

Resource: VirtualIP (class=ocf provider=heartbeat type=IPaddr2)

Attributes: ip=192.168.5.25 cidr_netmask=24

Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)

stop interval=0s timeout=20s (VirtualIP-stop-timeout-20s)

monitor interval=10s timeout=20s (VirtualIP-monitor-interval-10s)

monitor interval=20s (VirtualIP-name-monitor-interval-20s)

Resource: iscsiUnit (class=lsb type=iscsi)

Attributes: portal=192.168.5.110 target=iqn.2015-02.com.cluster:tgt1

Operations: start interval=0 timeout=20 (iscsiUnit-start-0)

stop interval=0 timeout=20 (iscsiUnit-stop-0)

monitor start-delay=0 interval=0 timeout=30 (iscsiUnit-monitor-0)

Resource: iscsi-mount (class=ocf provider=heartbeat type=Filesystem)

Attributes: device=/dev/sda1 directory=/mnt/storage fstype=ext4

Operations: start interval=0 timeout=120 (iscsi-mount-start-0)

stop interval=0 timeout=120 (iscsi-mount-stop-0)

monitor interval=5s timeout=20s (iscsi-mount-name-monitor-interval-5s-timeout-20s)

Resource: BaseDatos (class=lsb type=mysqld)

Operations: monitor interval=20s timeout=20s (BaseDatos-name-monitor-interval-20s-timeout-20s)

Stonith Devices:

Fencing Levels:

Location Constraints:

Ordering Constraints:

start VirtualIP then start iscsiUnit (kind:Mandatory) (id:order-VirtualIP-iscsiUnit-mandatory)

start iscsiUnit then start iscsi-mount (kind:Mandatory) (id:order-iscsiUnit-iscsi-mount-mandatory)

start iscsi-mount then start BaseDatos (kind:Mandatory) (id:order-iscsi-mount-BaseDatos-mandatory)

Colocation Constraints:

VirtualIP with iscsiUnit (score:INFINITY) (id:colocation-VirtualIP-iscsiUnit-INFINITY)

iscsiUnit with iscsi-mount (score:INFINITY) (id:colocation-iscsiUnit-iscsi-mount-INFINITY)

iscsi-mount with BaseDatos (score:INFINITY) (id:colocation-iscsi-mount-BaseDatos-INFINITY)

Cluster Properties:

cluster-infrastructure: cman

dc-version: 1.1.11-97629de

no-quorum-policy: ignore

stonith-enabled: false

Anexo E: Scripts personalizados

Los scripts a continuación se utilizaron para tomar las 96 muestras de los tiempos de respuesta ante el fallo de los recursos tomcat, mysql o a su vez para medir el tiempo que demoraban migrar los recursos de un nodo a otro, para las dos soluciones de clúster.

Script para monitorear tiempo de respuesta de Pacemaker ante la detención del servicio tomcat.

Dentro del lazo for se procedió a matar el servicio tomcat y a esperar un tiempo para que Pacemaker pudiera detectar que tal servicio no se estaba ejecutando y lo reinicie nuevamente. Finalmente se tomaron los datos de detección de fallo de tomcat y de reinicio del mismo, de los logs almacenados en el directorio /var/log para corosync.log y tomcat.log, respectivamente.

```
#! /bin/bash
#Matar recurso y medir tiempo en que se mata el recurso
cd /usr/local/tomcat6
for ((a=0; a <= 120; a++ ))
do
        date | cut -c12-19 >> /root/Documentos/result/KillRecurso.txt
        ./bin/shutdown.sh
        sleep 20
done
```

```
cd /root/Documentos/result
#Tiempo que detecta apagado
cat  /var/log/cluster/corosync.log | grep "Tomcat6_monitor_10000: not running" | cut -c8-16 >> DetectRecApagado.txt

#Tiempo de reinicio de tomcat
cat /var/log/tomcat.log | grep "start =" | cut -c12-19 >> ReinicioRecurso.txt
```

Script para monitorear tiempo de respuesta de RGManager ante la detención del servicio tomcat

Dentro del lazo for se procedió a matar el servicio tomcat y a esperar un tiempo para que RGManager pudiera detectar que tal servicio no se estaba ejecutando y lo reinicie nuevamente. Finalmente se tomaron los datos de detección de fallo de tomcat y de reinicio del mismo, de los logs almacenados en el directorio /var/log para rgmanager.log.

```
#! /bin/bash
cd /root/Documentos/result
for ((a=0; a <= 120; a++ ))
do
        date | cut -c12-19 >> KillRecurso.txt
        service tomcat stop
        sleep 100
done
```

```
#Tiempo que detecta apagado
cat /var/log/cluster/rgmanager.log | grep "Tomcat6 > Service Is Not Running" | cut -
c8-15 >> DetectRecApagado.txt

#Tiempo de reinicio de tomcat
cat /var/log/cluster/rgmanager.log | grep "Generating New Config File
/etc/cluster/tomcat-6" | cut -c8-15 >> ReinicioRecurso.txt
```

Script para monitorear tiempo de respuesta de Pacemaker ante la detención del servicio mysqld

Dentro del lazo for se procedió a matar el servicio mysql y a esperar un tiempo para que Pacemaker pudiera detectar que tal servicio no se estaba ejecutando y lo reinicie nuevamente. Finalmente se obtuvieron los tiempos de detección de fallo de mysql y de reinicio del mismo, de los datos almacenados en el log mysqld.log que se aloja en el directorio /var/log.

```
#! /bin/bash
# Matar el proceso mysqld
for ((a=0; a <= 120; a++ ))
do
        killall -TERM mysqld
        sleep 15
done
```

```
# Ubicarse en el directorio donde se almacenarán los resultados
cd /root/Documentos/result

# Se guarda en el fichero KillResource.txt el tiempo en el que se detectó que el
servicio se había parado.
cat /var/log/mysqld.log | grep "mysqld.pid ended" | cut -d' ' -f2>> KillRecurso.txt

# Se guarda en el fichero StartResource.txt el tiempo en el que mysql sería iniciado
nuevamente.
cat /var/log/mysqld.log | grep "Starting mysqld" | cut -d' ' -f2>> ReinicioRecurso.txt
```

Script para monitorear tiempo de respuesta de RGManager ante la detención del servicio mysqld

Dentro del lazo for se procedió a matar el servicio mysql y a esperar un tiempo para que RGManager pudiera detectar que tal servicio no se estaba ejecutando y lo reinicie nuevamente. Finalmente se obtuvieron los tiempos de detección de fallo de mysql y de reinicio del mismo, de los datos almacenados en los logs mysqld.log y rgmanager.log alojados en el directorio /var/log.

```
#! /bin/bash
for ((a=0; a <= 120; a++ ))
do
        killall -TERM mysqld
```

```
        sleep 70
done

# Ubicarse en el directorio donde se almacenarán los resultados
cd /root/Documentos/result
#Tiempo que detecta apagado
cat /var/log/mysqld.log | grep "Shutdown completed" | cut -d' ' -f2>> KillRecurso.txt
cat /var/log/cluster/rgmanager.log | grep "Service Is Not Running" | cut -c8-15 >>
DetectRecApagado.txt
#Tiempo de reinicio de mysqld
cat /var/log/mysqld.log | grep "Starting mysqld" | cut -d' ' -f2>> ReinicioRecurso.txt
```

Script para monitorear tiempo de respuesta de Pacemaker al migrar los recursos del Clúster de Servidores de Bases de Datos, manualmente.

Este script fue usado en el nodo activo del clúster, para mover los recursos entre los nodos, obteniendo el tiempo en el que se apagaba el servicio mysqld del log mysqld.log, para posteriormente compararlo con el tiempo que se iniciaba ese mismo servicio en el nodo pasivo.

```
#! /bin/bash
for ((a=0; a <= 120; a++ ))
do
        #Para migrar recursos a nodo1
        pcs cluster unstandby BasePacemaker
```

```
        pcs cluster standby basePacemaker2
        sleep 10

        #Para migrar recursos a nodo 2
        pcs cluster unstandby basePacemaker2
        pcs cluster standby BasePacemaker
        sleep 10
done

#Ubicar los resultados en la carpeta result
cd /root/Documentos/result
cat /var/log/mysqld.log | grep "Shutdown completed" | cut -d' ' -f2-3>> Apagado.txt
```

Script para monitorear tiempo de respuesta de Pacemaker al migrar los recursos del Clúster de Servidores Web, manualmente.

Este script fue usado en el nodo activo del clúster, para mover los recursos entre los nodos, obteniendo el tiempo en el que se apagaba el servicio tomcat del log tomcat.log, para posteriormente compararlo con el tiempo que se iniciaba ese mismo servicio en el nodo pasivo.

```
#! /bin/bash

for ((a=1; a <= 120; a++ ))
do
```

```
        #Para migrar recursos a nodo1
        pcs cluster unstandby WebPacemaker
        sleep 3
        pcs cluster standby webPacemaker2
        sleep 20

        #Para migrar recursos a nodo 2
        pcs cluster unstandby webPacemaker2
        sleep 3
        pcs cluster standby WebPacemaker
        sleep 20
done

#Ubicar los resultados en la carpeta result
cd /root/Documentos/result
cat /var/log/tomcat.log | grep "###" | cut -c12-19 >> Apagado.txt
```

Script para monitorear tiempo de respuesta de RGManager al migrar los recursos del Clúster de Servidores de Bases de Datos, manualmente.

Este script fue usado en el nodo activo del clúster, para mover los recursos entre los nodos, obteniendo el tiempo en el que se apagaba el servicio mysqld del log del servicio rgmanager.log, para posteriormente compararlo con el tiempo que se iniciaba ese mismo servicio en el nodo pasivo.

```
#! /bin/bash

for ((a=0; a <= 120; a++ ))
do
        date | cut -c12-19 >> OrdenMigracion.txt
        clusvcadm -r BaseDatos -m baseRGManager2
        sleep 20
        clusvcadm -r BaseDatos -m baseRGManager
        sleep 20
done

cat /var/log/cluster/rgmanager.log | grep "Service service:BaseDatos is stopped" | cut -c8-15 >> InicioMigracion.txt
```

Script para monitorear tiempo de respuesta de RGManager al migrar los recursos del Clúster de Servidores Web, manualmente.

Este script fue usado en el nodo activo del clúster, para mover los recursos entre los nodos, obteniendo el tiempo en el que se apagaba el servicio Web, del log del servicio rgmanager.log, para posteriormente compararlo con el tiempo que se iniciaba ese mismo servicio en el nodo pasivo.

```
#! /bin/bash
for ((a=0; a <= 120; a++ ))
do
```

```
        date | cut -c12-19 >> OrdenMigracion.txt
        clusvcadm -r Web -m webRGManager2
        sleep 20
        clusvcadm -r Web -m webRGManager
        sleep 20
done

cat /var/log/cluster/rgmanager.log | grep "Service service:Web is stopped" | cut -c8-
15 >> InicioMigracion.txt
```

Anexo F: Resultados de las Pruebas de Estudio

Tabla 16. Resultados del porcentaje de consumo de memoria RAM para la tecnología de virtualización Xen en el modo de paravirtualización.

	Técnicas de Stress	
	ab	**stress**
Promedio	56,03 %	79,55 %
Desviación estándar	2,26 %	10,60 %
Error estándar	0,23 %	1,08 %
Cuartil 1	54,14 %	71,12 %
Cuartil 3	57,99 %	89,01 %
Rango intercuartil	3,85 %	17,89 %
Límite inferior	42,60 %	17,46 %
Límite superior	69,53 %	142,68 %

Tabla 17. Resultados del porcentaje de consumo de CPU para la tecnología de virtualización Xen en el modo de paravirtualización.

	Técnicas de Stress			
	ab		**stress**	
	% usuario	**% sistema**	**% usuario**	**% sistema**
Promedio	19,28 %	46,81 %	16,28 %	83,72 %
Desviación estándar	1,72 %	4,46 %	0,80 %	0,80 %

Tabla 17. Resultados del porcentaje de consumo de CPU para la tecnología de virtualización Xen en el modo de paravirtualización (continuación)

	Técnicas de Stress			
	ab		stress	
	% usuario	% sistema	% usuario	% sistema
Error estándar	0,18 %	0,46 %	0,08 %	0,08 %
Cuartil 1	18,68 %	46,93 %	16,00 %	83,50 %
Cuartil 3	20,21 %	49,06 %	16,50 %	84,00 %
Rango intercuartil	1,53 %	2,13 %	0,50 %	0,50 %
Límite inferior	14,09 %	40,55 %	14,50 %	82,00 %
Límite superior	24,80 %	55,44 %	18,00 %	85,50 %

Tabla 18. Resultados del porcentaje de consumo de memoria RAM para la tecnología de virtualización Xen en el modo de virtualización completa

	Técnicas de Stress	
	ab	stress
Promedio	93,19 %	78,04 %
Desviación estándar	0,41 %	5,95 %
Error estándar	0,04 %	0,61 %
Cuartil 1	92,93 %	74,22 %
Cuartil 3	93,51 %	81,66 %

Tabla 18. Resultados del porcentaje de consumo de memoria RAM para la tecnología de virtualización Xen en el modo de virtualización completa (continuación)

	Técnicas de Stress	
	ab	**stress**
Rango intercuartil	0,58 %	7,44 %
Límite inferior	91,19 %	51,91 %
Límite superior	95,25 %	103,98 %

Tabla 19. Resultados del porcentaje de consumo de CPU para la tecnología de virtualización Xen en el modo de virtualización completa

	Técnicas de Stress			
	ab		**stress**	
	% usuario	**% sistema**	**% usuario**	**% sistema**
Promedio	19,24 %	54,89 %	18,68 %	80,93 %
Desviación estándar	1,97 %	3,34 %	1,27 %	1,40 %
Error estándar	0,20 %	0,34 %	0,13 %	0,14 %
Cuartil 1	18,37 %	54,46 %	18,00 %	80,50 %
Cuartil 3	20,88 %	56,68 %	19,18 %	81,61 %
Rango intercuartil	2,51 %	2,22 %	1,18 %	1,11 %
Límite inferior	10,83 %	47,79 %	14,48 %	77,16 %
Límite superior	28,42 %	63,35 %	22,70 %	84,95 %

Tabla 20. Resultados del porcentaje de consumo de memoria RAM para la tecnología de virtualización Hyper-V en el modo de virtualización completa

	Técnicas de Stress	
	ab	**stress**
Promedio	59,85 %	86,51 %
Desviación estándar	1,02 %	3,25 %
Error estándar	0,10 %	0,33 %
Cuartil 1	59,09 %	84,21 %
Cuartil 3	60,71 %	89,33 %
Rango intercuartil	1,63 %	5,12 %
Límite inferior	54,21 %	68,85 %
Límite superior	65,59 %	104,69 %

Tabla 21. Resultados del porcentaje de consumo de CPU para la tecnología de virtualización Hyper-V en el modo de virtualización completa

	Técnicas de Stress			
	ab		**stress**	
	% usuario	**% sistema**	**% usuario**	**% sistema**
Promedio	21,55 %	40,20 %	15,33 %	84,67 %
Desviación estándar	2,45 %	3,08 %	0,90 %	0,90 %
Error estándar	0,25 %	0,31 %	0,09 %	0,09 %

Tabla 21. Resultados del porcentaje de consumo de CPU para la tecnología de virtualización Hyper-V en el modo de virtualización completa (cont.)

	Técnicas de Stress			
	ab		stress	
	% usuario	% sistema	% usuario	% sistema
Cuartil 1	21,18 %	40,08 %	14,86 %	84,48 %
Cuartil 3	22,77 %	41,79 %	15,52 %	85,14 %
Rango intercuartil	1,59 %	1,72 %	0,66 %	0,66 %
Límite inferior	16,41 %	34,93 %	12,90 %	82,52 %
Límite superior	27,54 %	46,94 %	17,49 %	87,10 %

Tabla 22. Resultados del tiempo de respuesta del stack Pacemaker, CMAN y Corosync al realizar la técnica de inyección de fallo a nivel de servicios

	Clúster de servidores web	Clúster de servidores de bases de datos
Promedio	3,67 s	6,33 s
Desviación estándar	2,58 s	2,81 s
Error estándar	0,26 %	0,29 %
Cuartil 1	1,00 s	3,00 s
Cuartil 3	6,00 s	9,00 s

Tabla 22. Resultados del tiempo de respuesta del stack Pacemaker, CMAN y Corosync al realizar la técnica de inyección de fallo a nivel de servicios (continuación)

	Clúster de servidores web	Clúster de servidores de bases de datos
Rango intercuartil	5,00 s	6,00 s
Límite inferior	-14,00	-15,00
Límite superior	21,00 s	27,00 s

Tabla 23. Resultados del tiempo de respuesta del complemento de alta disponibilidad de Red Hat al realizar la técnica de inyección de fallo a nivel de servicios

	Clúster de servidores web	Clúster de servidores de bases de datos
Promedio	28,90 s	27,94 s
Desviación estándar	16,40 s	15,74 s
Error estándar	1,67 %	1,61 %
Cuartil 1	16,75 s	15,00 s
Cuartil 3	42,25 s	44,00 s
Rango intercuartil	25,50 s	29,00 s
Límite inferior	-59,75	-72,00
Límite superior	118,75 s	131,00 s

Tabla 24. Resultados del tiempo de recuperación vs tiempo de monitoreo en Pacemaker para la técnica de inyección de fallo a nivel de servicios en el clúster de servidores de bases de datos

	Tiempo de monitoreo		
	10s	**20s**	**60s**
Promedio	3,67 s	6,70 s	27,64 s
Desviación estándar	2,58 s	6,04 s	16,01 s
Error estándar	0,26 %	0,62 %	1,63 %
Cuartil 1	1,00 s	1,00 s	16,00 s
Cuartil 3	6,00 s	11,00 s	41,00 s
Rango intercuartil	5,00 s	10,00 s	25,00 s
Límite inferior	-14,00	-29,00	-59,00
Límite superior	21,00 s	41,00 s	116,00 s

Tabla 25. Resultados del tiempo de recuperación vs tiempo de monitoreo en Pacemaker para la técnica de inyección de fallo a nivel de servicios en el clúster de servidores web

	Tiempo de Monitoreo		
	10s	**20s**	**60s**
Promedio	6,33 s	14,49 s	29,60 s
Desviación estándar	2,81 s	5,79 s	17,36 s
Error estándar	0,29 %	0,59 %	1,77 %
Cuartil 1	3,00 s	11,00 s	15,00 s

Tabla 25. Resultados del tiempo de recuperación vs tiempo de monitoreo en Pacemaker para la técnica de inyección de fallo a nivel de servicios en el clúster de servidores web (continuación)

	Tiempo de monitoreo		
	10s	**20s**	**60s**
Cuartil 3	9,00 s	19,00 s	43,75 s
Rango intercuartil	6,00 s	8,00 s	28,75 s
Límite inferior	-15,00	-13,00	-71,25
Límite superior	27,00 s	43,00 s	130,00 s

Tabla 26. Resultados del tiempo de respuesta del stack Pacemaker, CMAN y Corosync al realizar la técnica de inyección de fallo a nivel de nodos

	Clúster de servidores web	**Clúster de servidores de bases de datos**
Promedio	9,43 s	10,17 s
Desviación estándar	0,75 s	0,95 s
Error estándar	0,08 %	0,10 %
Cuartil 1	9,00 s	10,00 s
Cuartil 3	10,00 s	11,00 s
Rango intercuartil	1,00 s	1,00 s
Límite inferior	6,00 s	7,00 s
Límite superior	13,00 s	14,00 s

Tabla 27. Resultados del tiempo de respuesta del stack Pacemaker, CMAN y Corosync al realizar la migración de recursos de forma manual

	Clúster de servidores web	Clúster de servidores de bases de datos
Promedio	2,60 s	3,58 s
Desviación estándar	0,49 s	1,50 s
Error estándar	0,05 %	0,15 %
Cuartil 1	2,00 s	2,00 s
Cuartil 3	3,00 s	5,00 s
Rango intercuartil	1,00 s	3,00 s
Límite inferior	-1,00	-7,00
Límite superior	6,00 s	14,00 s

Tabla 28. Resultados del tiempo de respuesta del complemento de alta disponibilidad de RedHat al migrar los recursos de forma manual

	Clúster de servidores web	Clúster de servidores de bases de datos
Promedio	20,42 s	20,14 s
Desviación estándar	0,52 s	0,35 s
Error estándar	0,05 %	0,04 %
Cuartil 1	20,00 s	20,00 s

Tabla 28. Resultados del tiempo de respuesta del complemento de alta disponibilidad de RedHat al migrar los recursos de forma manual (continuación)

	Clúster de servidores web	**Clúster de servidores de bases de datos**
Cuartil 3	21,00 s	20,00 s
Rango intercuartil	1,00 s	0,00 s
Límite inferior	17,00 s	20,00 s
Límite superior	24,00 s	20,00 s

BIBLIOGRAFÍA

[1] Wikipedia, Virtualization, en.wikipedia.org

[2] Nazareno Gonzalo, Virtualización de Servidores, http://www.gonzalonazareno.org, fecha de publicación 2011.

[3] Punina Carina, Virtualización vs Paravirtualización, https://prezi.com/uooqr3yql22u/virtualizacion-vs-paravirtualizacion/, fecha de publicación Agosto 2011.

[4] Red Hat, Virtualización Integrada, http://www.redhat.com/f/pdf/virtualization/

[5] Jones Kathryn y González Esteban, Secretos del VM: Virtualización y Drivers – pdf.

[6] Funes Javier, Virtualizando con Xen –pdf.

[7] Talens Sergio, Herramientas de Virtualización Libres para sistemas GNU/Linux – pdf, fecha de publicación 2008.

[8] SearchDataCenter, Virtualización basada en contenedores, http://searchdatacenter.techtarget.com/es/definicion/virtualizacion-basada-en-contenedores-virtualizacion-a-nivel-de-sistema-operativo, fecha de publicación abril del 2014.

[9] Alegsa, Definición de Virtualización a nivel de sistema operativo, http://www.alegsa.com.ar/

[10] Facultad de Informática de Barcelona, Virtualización de aplicaciones, http://inlab.fib.upc.edu/es/virtualizacion-de-aplicaciones, fecha de publicación 2011.

[11] Microsoft, Virtualización, http://www.microsoft.com/spain/virtualizacion/

[12] OpenSystensConsulting, Entendiendo la virtualización de escritorios, http://www.opensystemsconsulting.es, fecha de publicación 2011

[13] Microways, Introducción a la Virtualización, http://www.mikroways.net/2009/09/25/introduccion-a-la-virtualizacion/, fecha de publicación 25 de septiembre.

[14] CentOS, Virtualización en CentOS, http://wiki.centos.org/es/HowTos/Virtualization/, fecha de publicación diciembre del 2007.

[15] Periáñez Francisco, Tutorial de VirtualBox, http://fpg.hol.es/VirtualBox/

[16] Linux Journal, Xen Virtualization and Linux Clustering part1, http://www.linuxjournal.com/article/8812, fecha de publicación enero del 2006.

[17] Cujano Silvia, Análisis e implementación de alta disponibilidad mediante clustering en Sistemas de Call Center basados en VoIP –pdf, fecha de publicación 2011.

[18] RedHat, https://access.redhat.com/

[19] EcuRed, Cluster de Alta Disponibilidad, http://www.ecured.cu/

[20] Microsoft, Recursos de un clúster de servidores, http://msdn.microsoft.com/, fecha de publicación enero del 2005.

[21] Clavijo Paulo, Clúster de Alta Disponibilidad (HA), http://www.lintips.com/?q=node/119, fecha de publicación 2010.

[22] vtiger, Vtiger , www.vtiger.com

[23] MRBS, MRBS Introduction, http://mrbs.sourceforge.net/

[24] Kernelthread, An Introduction to virtualization, http://www.kernelthread.com/publications/virtualization/, fecha de publicación enero del 2004.

[25] Wiki Debian, Xen, https://wiki.debian.org/es/Xen, fecha de publicación agosto del 2012.

[26] Reguera Alain, Creando e instalando una instancia domU de CentOS 5, http://wiki.centos.org/es/HowTos/Xen/, fecha de publicación diciembre del 2007.

[27] Linux Foundation, http://wiki.xen.org/wiki/

[28] JJVelasco (2010 Noviembre 11), Virtualización “low cost” con Citrix XenServer, http://bitelia.com/2010/11/virtualizacion-low-cost-con-citrix-xenserver

[29] Herrero Héctor, Citrix XenServer, http://www.bujarra.com/ProcedimientoCitrixXenServer.html

[30] Citrix, XenServer, http://lac.citrix.com/products/xenserver/tech-info.html

[31] User Mode Linux, The User-mode Linux Kernel Home Page, http://user-mode-linux.sourceforge.net/

[32] VMware, Hoja de Datos VMware vSphere, Ediciones de Enterprise y Enterprise Plus – pdf

[33] VMware, http://www.vmware.com/

[34] Microsoft, Windows Server Introducción a Hyper-V, https://technet.microsoft.com/

[35] Hwang Jinho, Zeng Sai and Frederick y Wu, Wood Thimoty, A Component-Based Performance Comparison of Four Hypervisors, http://ieeexplore.ieee.org/xpls/, fecha de publicación 2013

[36] Chierici Andrea y Veraldi Riccardo, A quantitative comparison between xen and kvm, http://iopscience.iop.org, fecha de publicación 2010

[37] Microsoft, Lo Nuevo en “Windows Server 2012 Failover Clustering” Parte 1, http://blogs.technet.com/

[38] HighAvailability (2010 Enero 25), Heartbeat, http://www.linux-ha.org/wiki/Heartbeat, fecha de publicación enero del 2010.

[39] Corosync, Corosync – The Corosync Cluster Engine, http://corosync.github.io/corosync/

[40] CMAN, CMAN Project Page, http://www.sourceware.org/cluster/cman/

[41] Pacemaker, Pacemaker - A Scalable High Availability cluster resource manager, http://clusterlabs.org/

[42] SupremeIndia, HP Pro 6300MT 3rd Gen, http://www.supremeindia.com/products/

[43] PCEL, Switch HP V1905-24, http://pcel.com/HP-JD990A-85403.

[44] VMware, VMware Compatibility Guide, http://www.vmware.com/resources/compatibility/

[45] Microsoft, Instalar el rol Hyper-V y configurar una máquina virtual, https://technet.microsoft.com/, fecha de publicación agosto del 2012.

[46] Linux Foundation, Fedora Host Installation, http://wiki.xen.org/wiki/Fedora_Host_Installation, fecha de publicación noviembre del 2014.

[47] Linux Foundation, DomU Install with Virt-Manager, http://wiki.xen.org/wiki/DomU_Install_with_Virt-Manager, fecha de publicación octubre del 2014.

[48] DRBD, DRBD Software Development for High Availability Clusters, http://drbd.linbit.com/, fecha de publicación 2014

[49] Pacemaker, Pacemaker A High Availability cluster resource manager, http://clusterlabs.org/quickstart.html

[50] RedHat, Capítulo 3. RGManager, https://access.redhat.com/, fecha de publicación 2015.