# ADMINISTRACIÓN, MANTENIMIENTO, CONFIGURACIÓN Y MONITOREO DE LOS EQUIPOS SERVIDORES DEL GRUPO GID-CONUSS CON ÉNFASIS EN EL ANÁLISIS Y REESTRUCTURACIÓN DE LOS MODELOS DE ALTA DISPONIBILIDAD Y COMPUTACIÓN EN LA NUBE.

JOHN EDINSON LIZARAZO TORRES
DIEGO ALBERTO NOGUERA GIRALDO

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2013

# ADMINISTRACIÓN, MANTENIMIENTO, CONFIGURACIÓN Y MONITOREO DE LOS EQUIPOS SERVIDORES DEL GRUPO GID-CONUSS CON ÉNFASIS EN EL ANÁLISIS Y REESTRUCTURACIÓN DE LOS MODELOS DE ALTA DISPONIBILIDAD Y COMPUTACIÓN EN LA NUBE.

JOHN EDINSON LIZARAZO TORRES
DIEGO ALBERTO NOGUERA GIRALDO

Trabajo de grado para optar al título de

Ingeniero de Sistemas

Director
MSc. Manuel Guillermo Flórez Becerra

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2013

## DEDICATORIA

"*Es muy difícil saber todo lo que hace falta, y mucho más difícil ignorar todo lo que hace falta ignorar*"

**Noel Clarasó.**

A mis padres, Juan Carlos y Elizabeth por darme la posibilidad de iniciar hace ya mucho tiempo este sueño y quienes con su apoyo moral y espiritual acompañaron todo este proceso.

A mis hermanos menores, Daniel, Andrés y Sebastián por recordarme que siempre hay una familia dispuesta a apoyar la construcción de un sueño.

A mi gran amigo, colega y compañero Diego Noguera que aceptó el reto y tuvo el honor de acompañar y enfrentar cada etapa, obstáculo y proceso. De todo corazón, mil gracias.

A mis amigos incondicionales, Alex, Lizeth, Emmanuel, Willian y Silvia que con su apoyo y maravillosos consejos iluminaron mi camino hacia el éxito.

Por último pero no menos importante, a todos mis amigos, conocidos y familiares quienes aportaron un granito de arena para alcanzar este logro.

**John Edinson Lizarazo Torres**

## DEDICATORIA

“La vida es una serie de colisiones con el futuro; no es una suma de lo que hemos sido, sino de lo que anhelamos ser”.

**José Ortega y Gasset**

A Dios, por darme la sabiduría y el entendimiento suficiente para sacar adelante mi grado profesional.

A mis padres Julio y Rocío por su apoyo incondicional y acompañamiento durante toda la carrera y por brindarme la posibilidad de culminar mi estudio de pregrado satisfactoriamente.

A mi compañero y amigo John Lizarazo, por su gran apoyo durante el desarrollo de este proyecto, por su paciencia y contribución en mi crecimiento personal.

A mis hermanos Andrés y Alejandra por sus lecciones de vida, enseñanza e instrucción como hermanos mayores.

Finalmente y no menos importante a mis demás familiares y amigos con los cuales compartí momentos memorables e inolvidables.

**Diego Alberto Noguera Giraldo**

## AGRADECIMIENTOS

Al profesor Manuel Guillermo Flórez Becerra por la confianza depositada en nosotros y por guiar con gran sabiduría y conocimientos el trabajo realizado.

A la fundación Raúl D. Ocazionez por apoyar e impulsar la labor investigativa de la Escuela de Ingeniería de Sistemas e Informática.

Al Grupo de Investigación y Desarrollo en Computación en la Nube, Seguridad, Servidores y Servicios, por permitirnos adquirir conocimientos de alto valor agregado y por proporcionar los recursos necesarios para la ejecución de este proyecto.

A la comunidad de software libre por hacer valiosos aportes en distintos campos para la puesta en marcha de cualquier tipo de proyecto o investigación.

A la Escuela de Ingeniería de Sistemas y en general a la Universidad Industrial de Santander por permitir espacios para la formación de personas con grandes cualidades éticas y profesionales.

CONTENIDO

INTRODUCCIÓN ..... 19
1. INFORMACIÓN DEL PROYECTO ..... 21
1.1. Descripción del problema ..... 21
1.2. Justificación ..... 22
1.3. Viabilidad ..... 22
2. OBJETIVOS ..... 23
2.1. Objetivo General ..... 23
2.2. Objetivos Específicos ..... 23
3. MARCO TEÓRICO ..... 25
3.1. Alta Disponibilidad ..... 25
3.1.1. Clúster de Alta Disponibilidad ..... 25
3.2. Virtualización ..... 29
3.2.1. Tipos de Virtualización ..... 29
3.2.2. Ventajas y Desventajas ..... 31
3.3. Computación en la Nube ..... 32
3.3.1. Definición ..... 32
3.3.2. Características Esenciales ..... 32
3.3.3. Modelos de Servicio ..... 34
4. ANÁLISIS Y DISEÑO ..... 36
4.1. Estudio de la situación actual ..... 36
4.2. Esquema general de la infraestructura ..... 38
4.3. MODELO DE COMPUTACIÓN EN LA NUBE ..... 39

4.3.1. Análisis general del modelo ....40
4.4. MODELO DE ALTA DISPONIBILIDAD ....42
4.4.1. Análisis general del modelo ....44
4.5. Consideraciones iniciales de diseño ....46
4.6. Diseño del modelo ....47
4.6.1. Diseño del Modelo de alta disponibilidad ....48
4.6.2. Diseño del Modelo de computación en la nube ....49
4.6.3. Modelo de soporte para los sistemas en producción ....50
4.6.4. Modelo de balanceo y distribución de recursos ....51
5. IMPLEMENTACIÓN DEL MODELO ....53
5.1. Modelo de Servicio IaaS ....54
5.2. Herramientas Alta Disponibilidad ....55
5.2.1. DRBD (Distributed Replicated Block Device) ....55
5.2.2. Corosync/OpenAIS ....57
5.2.3. Pacemaker ....57
5.3. Selección Herramienta Datos Compartidos en Red ....59
5.4. Selección Herramienta de Sincronización ....61
5.5. Selección Herramienta Balanceo de Carga ....62
5.6. Implementación modelo de alta disponibilidad ....64
5.7. Implementación modelo computación en la nube ....66
5.8. Implementación modelo de balanceo y distribución de recursos ....68
5.9. Implementación del sistema de soporte ....70
5.10. Migración de servicios ....70
6. PUESTA EN MARCHA DEL MODELO ....72

6.1. Servicios Virtualizados ..... 72
6.1.1. Biblioteca de imágenes de servidores virtuales ..... 72
6.2. Copias de Seguridad ..... 75
6.2.1. Copia de seguridad del sistema ..... 75
6.2.2. Copia de seguridad de máquinas virtuales ..... 76
6.2.3. Copia de seguridad de los archivos de configuración ..... 76
6.2.4. Copia de seguridad de los datos ..... 77
6.3. Seguridad ..... 78
6.3.1. Actualizaciones de seguridad ..... 78
6.3.2. Sistemas de control y monitoreo ..... 79
6.4. Documentación administrativa ..... 80
6.5. Pruebas de funcionamiento ..... 81
6.5.1. Pruebas de instanciación de imágenes virtuales ..... 81
6.5.2. Pruebas de migración de instancias ..... 86
6.5.3. Pruebas de migración de máquinas virtuales en alta disponibilidad ..... 89
6.5.4. Pruebas del sistema de balanceo de carga ..... 93
7. CONCLUSIONES ..... 96
8. RECOMENDACIONES ..... 98
BIBLIOGRAFIA ..... 100
ANEXOS ..... 102

LISTA DE TABLAS

Tabla 1: Comparación herramientas datos compartidos en red.....59
Tabla 2. Tiempo de duración del proceso de instanciación. .....83
Tabla 3. Tiempos de recuperación de la disponibilidad del servicio .....87
Tabla 4. Tiempos de recuperación de la comunicación a través de ping.....90
Tabla 5. Tiempos de recuperación del frontend.....91

LISTA DE FIGURAS

Figura 1. Clúster de Alta Disponibilidad de dos Nodos. .....26
Figura 2. Esquema de clúster con infraestructura Activo/Pasivo. .....27
Figura 3. Esquema de clúster con infraestructura Activo/Activo. .....28
Figura 4. Esquema de clúster con infraestructura N+1. .....28
Figura 5. Esquema de clúster con infraestructura Split-site. .....29
Figura 6. Tipos de Hipervisor. .....31
Figura 7. Modelos de Servicio. .....35
Figura 8. Esquema General de la Infraestructura. .....38
Figura 9. Modelo de Prestación de Servicios para la EISI. .....39
Figura 10. Modelo de Alta Disponibilidad Anterior. .....43
Figura 11. Esquema base de diseño. .....47
Figura 12. Diseño del Modelo de Alta Disponibilidad. .....48
Figura 13. Diseño del Modelo de computación en la nube. .....49
Figura 14. Modelo de soporte para los sistemas en producción. .....50
Figura 15. Diseño del Modelo de balanceo y distribución de recursos. .....52
Figura 16. Interfaz de administración de OpenNebula. .....54
Figura 17. Logotipo DRBD. .....55
Figura 18. Logotipo Corosync. .....57
Figura 19. Logotipo Pacemaker. .....57
Figura 20. Volúmenes Distribuidos. .....61
Figura 21. Volúmenes Replicados. .....62
Figura 22. Interfaz de administración del ZEN Load Balancer. .....63
Figura 23. Modelo de alta disponibilidad actual. .....65
Figura 24. Modelo de computación en la nube actual. .....67
Figura 25. Modelo de balanceo y distribución de recursos actual. .....69
Figura 26. Esquema de respaldos periódicos locales. .....77
Figura 27. Esquema jerárquico de documentación. .....81
Figura 28. Cuadro distribución de usuarios. .....93

Figura 29. Carga de red y peticiones sobre una de las máquinas con el sistema de balanceo. .....94
Figura 30. Carga de red y peticiones sobre una de las máquinas sin el sistema de balanceo. .....94

LISTA DE ANEXOS

Anexo A: Prueba del sistema de balanceo de carga……92

RESUMEN

**TÍTULO:** ADMINISTRACION, MANTENIMIENTO, CONFIGURACIÓN Y MONITOREO DE LOS EQUIPOS SERVIDORES DEL GRUPO GID-CONUSS CON ENFASIS EN EL ANÁLISIS Y REESTRUCTURACIÓN DE LOS MODELOS DE ALTA DISPONIBILIDAD Y COMPUTACION EN LA NUBE.*

**AUTORES:** LIZARAZO TORRES, John Edinson**

NOGUERA GIRALDO, Diego Alberto**

**PALABRAS CLAVE:** Virtualización, Alta disponibilidad, Computación en la nube, IaaS, Diseño Modular.

**DESCRIPCIÓN:** El diseño de infraestructuras de IT abarca la distribución lógica y ordenada tanto de componentes de hardware como de software, buscando maximizar el uso de los recursos a disposición con el fin de crear entornos de IT adecuados para el hospedaje, desarrollo y ejecución de aplicaciones con cierto valor agregado tanto para los dueños del negocio como para los usuarios finales.

Con la aparición de nuevas tecnologías en este campo de aplicación, surgen nuevos paradigmas que traen consigo esquemas más flexibles y eficientes de implementación, optimizando el uso de recursos, recortando costos operativos, reduciendo la complejidad de la labor administrativa y abriendo puertas a nuevas opciones de extensión y actualización de la infraestructura, acordes con los intereses del negocio.

En el presente trabajo se hace uso de tecnologías emergentes como la virtualización y estilos de diseño provenientes de la ingeniería del software como el diseño basado en componentes, con el fin de proponer un sistema modular de fácil manejo para la arquitectura en producción propiedad del grupo GID-CONUSS, la cual ofrece un conjunto de servicios que benefician a estudiantes, profesores y grupos de investigación de la comunidad académica de la Universidad Industrial de Santander. Este objetivo se logra mediante la redefinición de sus sistemas base de alta disponibilidad y de computación en la nube en un proceso que busca establecer puntos de mejora integrando los enfoques anteriormente mencionados y agregando nueva funcionalidad que brinde un mejor servicio al usuario.


* Trabajo de Grado en la Modalidad de Investigación.

** Facultad de Ingenierías Físico Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director MSc. Manuel Guillermo Flórez Becerra

ABSTRACT

**TITLE:** ADMINISTRATION, MAINTENANCE, SETTING AND MONITORING FOR GID-CONUSS GROUP'S SERVER INFRASTRUCTURE WITH EMPHASIS ON ANALYSIS AND RESTRUCTURING OF HIGH AVAILABILITY AND CLOUD COMPUTING MODELS.*

**AUTHORS:** LIZARAZO TORRES, John Edinson**

NOGUERA GIRALDO, Diego Alberto**

**KEYWORDS:** Virtualization, High Availability, Cloud Computing, IaaS, Modular Design.

**DESCRIPTION:** The IT infrastructure design cover logical and orderly distribution of both hardware and software. It seeks to maximize the use of available resources in order to create suitable IT environments for hosting, developing and running of value-added applications for both business owners and end users. The emergence of new technologies on this field of application has created new paradigms with more flexible and efficient implementation schemes, optimizing use of resources, cutting operating, reducing costs, reducing the complexity of the administrative work and opening doors to new extension and upgrading options, consistent with the interests of the business.

In our present work, we make use of emerging technologies such as virtualization and design styles from software engineering like component-based design, in order to propose a modular and easy to use system for the GID-CONUSS group's production architecture, which houses a collection of services that benefit students, teachers and research groups from Universidad Industrial de Santander's academic community. This is achieved by redefining their base high availability and cloud computing systems in a process that seeks to establish areas for improvement. These goals will be accomplished by integrating approaches mentioned above and adding new functionality to provide better service to the user.

* Undergraduate final project, research modality.
** Physico-Mechanical Engineering Faculty. Systems Engineering and Computer Science School. Director MSc. Manuel Guillermo Flórez Becerra

# INTRODUCCIÓN

A través de los años se han buscado nuevas formas de diseñar los sistemas, estableciendo nuevos esquemas que permitan llevar a cabo procesos de una forma más segura, rápida y confiable. A la vez, siempre ha existido cierta preocupación por mantener actualizados los sistemas actuales de la organización a la vanguardia de las nuevas tecnologías.

Los diseños de entornos de IT no siempre son construidos tomando en cuenta propiedades como la escalabilidad y la actualización hacia nuevos sistemas de servicio y control; o tal vez sí, pero a veces el enfoque no es el adecuado en concordancia con intereses como la reducción del tiempo y costo de mantenimiento. De forma análoga al desarrollo del software, muchas veces las reparaciones y mejoras se hacen sobre el código fuente, o en nuestro caso, sobre los archivos de configuración principales, realizando en el proceso modificaciones en gran o menor medida sobre cada una de las piezas esenciales del sistema, todo ello con el fin de adaptarlos a lo que se quiere lograr. El problema de este tipo de enfoques es que son muy complejos y costosos de mantener.

Una solución a este tipo de dificultades viene dada en la división del sistema en pequeños subsistemas que sean lo suficientemente independientes unos de otros, separando la funcionalidad en tareas y procesos pequeños. Este es el panorama que quiere mostrar la arquitectura basada en componentes. Se divide el diseño en componentes lógicos y funcionales, exponiendo una interfaz de comunicación para servir a sus pares o al sistema en general [2]. Debido a la forma en que están construidos, cuentan con un grado mayor de flexibilidad frente a sistemas clásicos. Esta ventaja se ve reflejada en la distribución y aprovechamiento de los recursos, seguridad, confiabilidad y manejabilidad.

Los sistemas basados en virtualización abstraen el concepto de la arquitectura basada en componentes, de forma similar a como se aplica en el campo de desarrollo del software. Debido a que los servicios no están arraigados a la capa

de hardware y se pueden aislar dentro de entornos con ambientes simulados, es posible subdividir la arquitectura en pequeños servicios cuya capa externa (aquella que está a la vista del usuario) es igual o muy similar al escenario en el que el servicio completo se implementa directamente sobre la capa de hardware.

El presente trabajo, utiliza elementos como la virtualización y el diseño basado en componentes, con el fin proponer un sistema modular para la arquitectura en producción propiedad del grupo CONUSS, que alberga un conjunto de servicios de los cuales se benefician estudiantes y profesores de la comunidad académica. Se quiere lograr este objetivo mediante la redefinición de sus sistemas base de alta disponibilidad y de computación en la nube integrando los enfoques anteriormente mencionados.

# 1. INFORMACIÓN DEL PROYECTO

## 1.1. Descripción del problema

El grupo GID-CONUSS (Grupo de Investigación y Desarrollo en Computación en la Nube, Seguridad, servidores y servicios) actualmente destina tiempo y recursos en diseñar e implementar esquemas y servicios de tipo cliente servidor. Cuenta con una infraestructura que soporta los sistemas base para el desarrollo y puesta en marcha de los productos e investigaciones creadas en el grupo [1],[3], así como los servicios prestados a la comunidad académica que surgen en base a estos.

El esquema cuenta con dos piezas principales: Un sistema de alta disponibilidad (HA) que brinda una capa de tolerancia a fallos que impide el corte no programado y no anticipado de servicios característicos del grupo como las aulas virtuales MEIWEB y MOODLE. La otra pieza es un sistema de computación en la nube (CC) que administra la creación, configuración y mantenimiento de sistemas virtuales para otorgar a los usuarios, un servicio para la adecuación de plataformas con miras al desarrollo e implementación de proyectos de grado.

Los sistemas son independientes uno del otro, aunque funcionan sobre la misma arquitectura. Ante el deseo del grupo CONUSS de ampliar la infraestructura con el fin de albergar más servicios para sus usuarios y soportar nuevas actividades de investigación, se presenta un punto de quiebre que impide escalar los sistemas actuales a nuevos equipos. Una de las mayores razones de esta problemática, es la dependencia funcional que tienen los esquemas actuales del hardware y componentes de software del equipo y de la poca compatibilidad, en el caso del sistema de HA, con entornos de más de 2 equipos servidores. Además se pueden sumar la debilidad parcial que tiene el sistema de computación en la nube de actuar frente a fallos, esto debido a no contar con características de control y de no interactuar con el sistema de HA actual.

## 1.2. Justificación

Ante la incorporación de un nuevo equipo servidor a la infraestructura que se encuentra en producción, se hace necesario estudiar los modelos de alta disponibilidad y computación en la nube con el fin de analizar las posibilidades de integración. Además, con miras a establecer un sistema que soporte la extensión a nuevo hardware, servicios y actividades futuras de investigación, deben aplicarse modificaciones y correcciones sobre los sistemas, creando en el proceso un modelo que sea muy flexible y escalable.

## 1.3. Viabilidad

Con la adquisición de nuevo hardware por parte del grupo CONUSS, gracias a una donación de la fundación Raul Ocazionez, se abren nuevas puertas para la investigación en diversas ramas en las cuales actualmente se llevan a cabo proyectos. Los equipos están a disposición y preparados para ser configurados en base al planteamiento que se quiere desarrollar. Además de un equipo servidor, también se adquirieron equipos de cómputo de altas prestaciones con el fin de crear un entorno de pruebas que soporte la puesta en marcha de proyectos con miras a implementar componentes y servicios para la infraestructura en producción. Gracias a esto, es posible validar cada uno de los modelos y componentes durante la fase de desarrollo del proyecto sin la necesidad de interrumpir o afectar los procesos en el entorno de producción. En base a que el proyecto tiene enfoque que parte de tecnologías como la virtualización y de principios de desarrollo de software como la arquitectura basada en componentes, se facilita aún más crear escenarios de prueba para la instalación y configuración de sistemas y paquetes de software con cierto grado de utilidad para el presente proyecto.

# 2. OBJETIVOS

## 2.1. Objetivo General

Administrar, configurar, monitorear y asegurar el funcionamiento de la organización actual de servidores del grupo GID-CONUSS, enfatizando en la creación de un entorno virtualizado, con el fin de mejorar y consolidar los modelos de alta disponibilidad y computación en la nube.

## 2.2. Objetivos Específicos

- Continuar con el proceso actual de administración, mantenimiento y monitoreo de los servidores, clúster de alta disponibilidad y servicios de computación en la nube.
  - Monitoreo del sistema.
  - Realizar tareas de mantenimiento programadas y Copias de seguridad.
  - Recuperación del sistema o servicios en caso de fallas.
  - Realizar tareas relacionadas con la atención a los usuarios de los diferentes servicios.

- Configurar un entorno de pruebas con equipos disponibles en el grupo GID-CONUSS, que permita realizar actividades como:

  - Optimizar, asegurar y validar las configuraciones actuales de los módulos de alta disponibilidad, computación en la nube y los sistemas en general.
  - Emular escenarios de fallo de los sistemas para que no se presenten en los servidores en producción.
  - Probar y validar nuevos módulos y servicios que vayan a ser incorporados al entorno en producción.

- Estudiar los modelos de alta disponibilidad y computación en la nube, con el fin de proponer e implementar un modelo que soporte la incorporación de un nuevo nodo al clúster en producción, que funcionará como respaldo de servicios virtuales sensibles y apoyo de los modelos mencionados anteriormente.

- Actualizar y/o mejorar los manuales de administración, automatización de tareas y las políticas de seguridad.

- Entrenar y asesorar los relevos administrativos para garantizar la continuidad de los procesos.

# 3. MARCO TEÓRICO

Para entender y comprender este proyecto, desde su enfoque hasta su funcionamiento es necesario tener en cuenta distintos conceptos, como lo son alta disponibilidad, infraestructura como servicios y virtualización. Es por ello que los abordaremos en este apartado.

## 3.1. Alta disponibilidad

Tomando la definición expuesta por el IEEE TFCC[1], alta disponibilidad se refiere a "*la disponibilidad de los recursos en un sistema computacional en el evento de fallo de componentes del sistema. Esto se puede lograr de varias maneras, abarcando todo el espectro que va desde un extremo con soluciones que utilizan hardware personalizado y redundante para garantizar la disponibilidad, hasta otro extremo con soluciones que ofrecen las soluciones de software que utilizan componentes de hardware genericos*."[10]

### 3.1.1. Clúster de alta disponibilidad

Un clúster de alta disponibilidad es una arreglo de 2 o más equipos servidores que están continuamente monitoreándose entre sí. Comparten ciertos recursos y reaccionan frente a eventos inesperados aplicando acciones que aseguran la disponibilidad de los servicios. La finalidad de este tipo de clúster es eliminar puntos de fallo mediante redundancia en niveles como hardware, el almacenamiento, la red, las aplicaciones etc.

[1] IEEE Task Force on Cluster Computing

Existe una gran diferencia, para evitar confusiones, entre clúster de alta disponibilidad y clúster de alto rendimiento. El segundo es una configuración de equipos diseñado para proporcionar capacidades de cálculo mayores que la que proporcionan los equipos individuales, mientras que el primer tipo de clúster está diseñado para garantizar el funcionamiento ininterrumpido de ciertas aplicaciones.

El siguiente esquema presenta un modelo de clúster de alta disponibilidad con dos nodos (equipos servidores), donde muestra el funcionamiento básico del clúster.

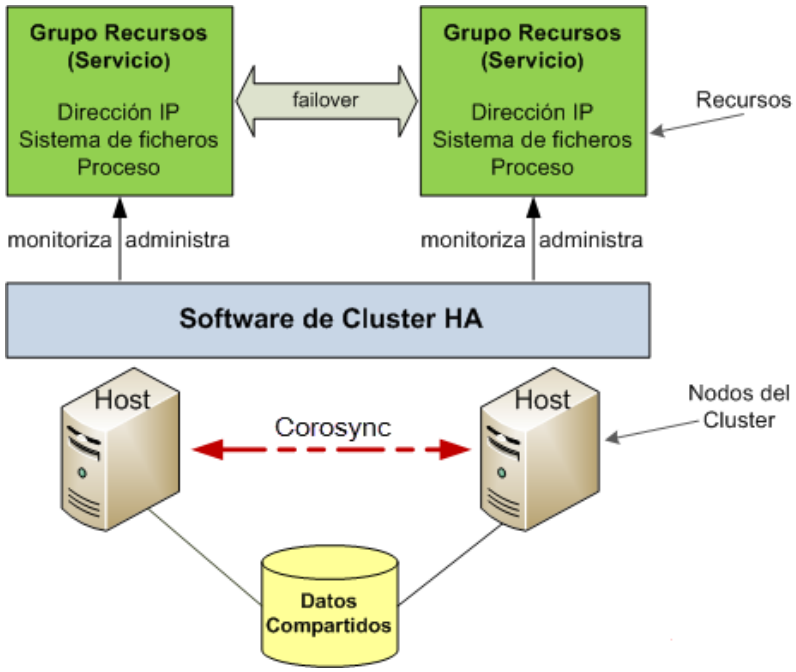



*Figura 1. Clúster de Alta Disponibilidad de dos Nodos. Fuente: http://www.lintips.com/*

En la figura se observan dos nodos (Hosts), los cuales comparten datos que se encuentran dentro de un disco o partición específica, la cual permite una sincronización constante y correcta. Además de los datos, se puede configurar y asegurar la disponibilidad de los recursos (servicios) instalados en ambos nodos. Los datos y los recursos se sincronizan a través del software de gestión del clúster y del uso de un demonio de comunicación que pasa información constantemente entre los nodos del clúster acerca del estado de los servicios y recursos compartidos.

La infraestructura base del clúster se puede implementar de diferentes maneras y con diferente software según sea el propósito o la disposición requerida de los recursos. A continuación se muestra algunas de las infraestructuras más utilizadas en el diseño de un clúster de alta disponibilidad utilizando herramientas como DRBD[2], Pacemaker[3], Corosync/OpenAIS[4].

- **Activo/Pasivo:** Uno de los nodos del clúster tiene activos los servicios y los sistemas de archivos compartidos, el otro, los mantiene en espera en caso de un fallo.

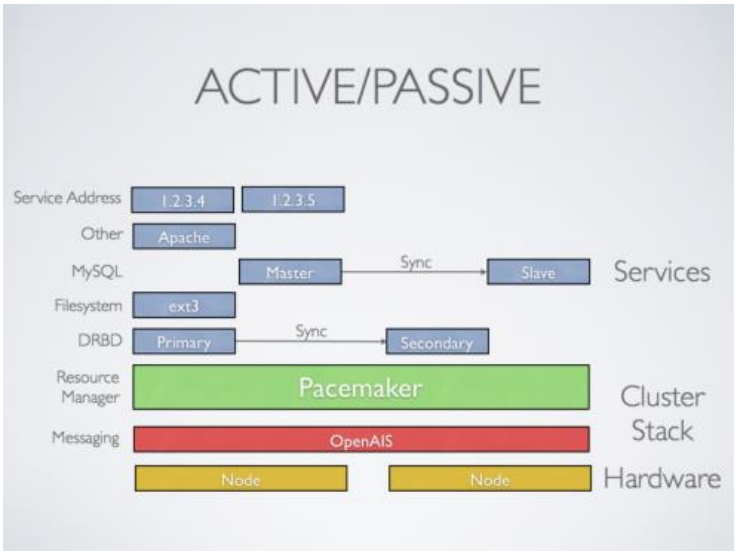



*Figura 2. Esquema de clúster con infraestructura Activo/Pasivo. Fuente: http://www.clusterlabs.org/*

- **Activo/Activo:** Todos los nodos del clúster se encuentran en estado activo, esto permite que todos los nodos sean utilizados potencialmente en caso de fallo, corriendo simultáneamente múltiples copias de los servicios y permitiendo así el balanceo de carga de trabajo entre los distintos nodos.

[2] Distributed Replicated Block Device. http://www.drbd.org/
[3] Scalable high-availability cluster resource manager. http://clusterlabs.org/
[4] OpenAIS is an open implementation of the Application Interface Specification (AIS) provided by the Service Availability Forum (SAForum or SA).

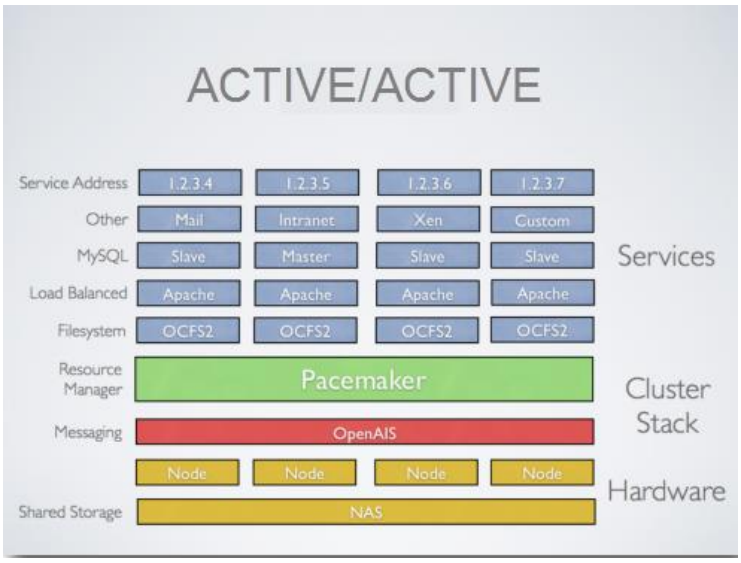



*Figura 3. Esquema de clúster con infraestructura Activo/Activo. Fuente: http://www.clusterlabs.org/*

- **N+1:** Esta infraestructura puede contar con más de dos nodos. Cuenta con varios nodos Activos/Pasivos, teniendo un nodo de backup común compartido, lo cual ayuda a reducir costos de hardware en los equipos.

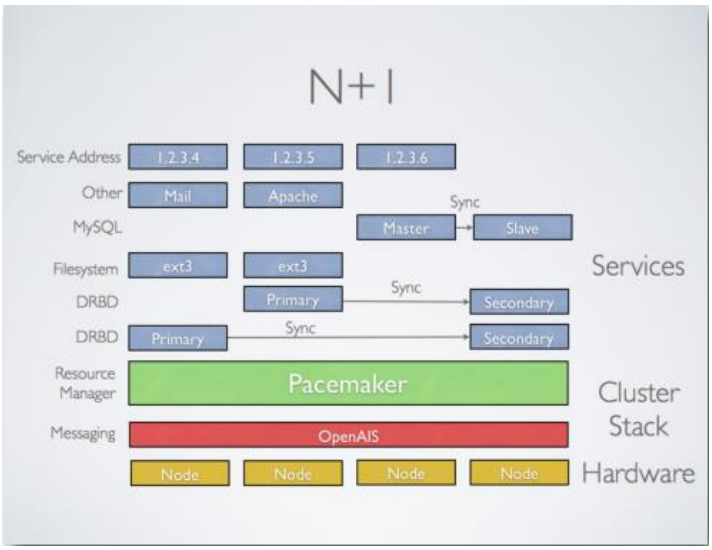



*Figura 4. Esquema de clúster con infraestructura N+1. Fuente: http://www.clusterlabs.org/*

- **Split-Site Clúster:** Un clúster de esta característica tiene compartido el almacenamiento de datos por red en los diferentes nodos del clúster, mientras que los servicios se encuentran activos en uno o más nodos y los omite en los otros.

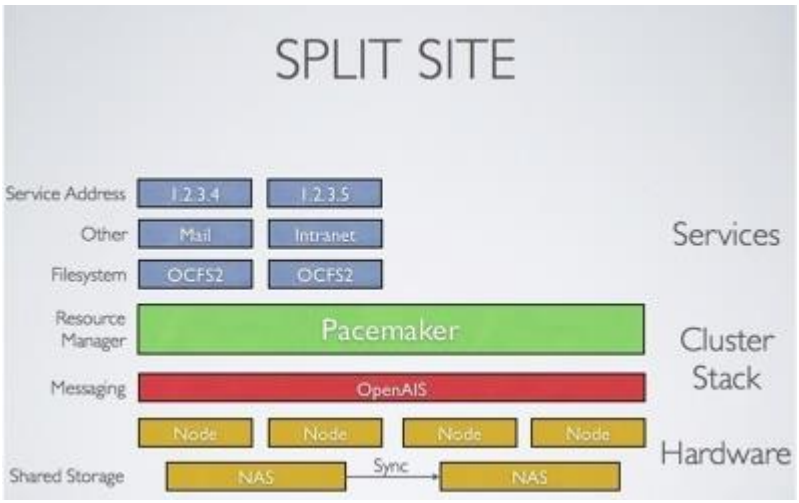



*Figura 5. Esquema de clúster con infraestructura Split-site. Fuente: http://www.clusterlabs.org/*

### 3.2. Virtualización

La virtualización es la puesta en marcha a través de software en una VMM[5] (Virtual Machine Monitor) de una versión virtual de algún recurso tecnológico, como puede ser una plataforma de hardware, un sistema operativo, un dispositivo de almacenamiento u otros recursos de red. Es decir, se crea una capa de abstracción entre el hardware de la máquina física (host), y el sistema operativo de la máquina virtual y allí los recursos se dividen entre uno o más entornos de ejecución.

#### 3.2.1. Tipos de virtualización

El concepto de virtualización permite su uso sobre múltiples recursos, para efectos de este proyecto se utilizó la virtualización por hardware, se ha conseguido por dos métodos distintos, virtualización parcial y virtualización completa.

[5]http://es.wikipedia.org/wiki/Hypervisor

**Paravirtualización** es una técnica en la cual el hipervisor presenta un API al sistema operativo invitado y éste realiza las peticiones a través de él al sistema operativo huésped, esto limita la cantidad de sistemas operativos virtualizables a solo aquellos que tengan soporte desde su núcleo.

**Virtualización parcial** es una técnica que simula parte del entorno de hardware, a diferencia del tipo de virtualización detallado a continuación. Es conocida como "Address Space Virtualization" debido a que simula espacios de direcciones. Se comparten recursos y procesos pero no es posible crear instancias separadas de sistemas operativos.

**Virtualización completa** es una técnica que se basa en traducción binaria para atrapar y virtualizar la ejecución de cierto tipo de instrucciones sensibles que no pueden ser virtualizadas, de esta manera las instrucciones críticas son reemplazadas por el hipervisor y emuladas por software.

Para controlar el tipo de virtualización usada se requiere de un monitor de máquina virtual conocido como hipervisor, el cual permite la ejecución de diferentes sistemas operativos en un mismo equipo.

Los hipervisores se clasifican en dos tipos: hipervisor tipo 1 (de primer nivel), el cual se ejecuta directamente sobre el hardware del equipo para controlar y administrar el sistema operativo invitado, este tipo de hipervisor representa la clásica implementación usada en arquitectura de máquinas virtuales.

Hipervisor tipo 2 (de segundo nivel), se ejecuta dentro del entorno de un sistema operativo convencional, de esta manera los sistemas operativos invitados se ejecutan a nivel de software separados del sistema operativo anfitrión por el hipervisor. Ejemplos de este tipo de hipervisor son Virtualbox[6] de Oracle y KVM[7].

[6] www.virtualbox.org
[7] www.linux-kvm.org/

La siguiente figura da una ilustración gráfica de los dos tipos de hipervisores.

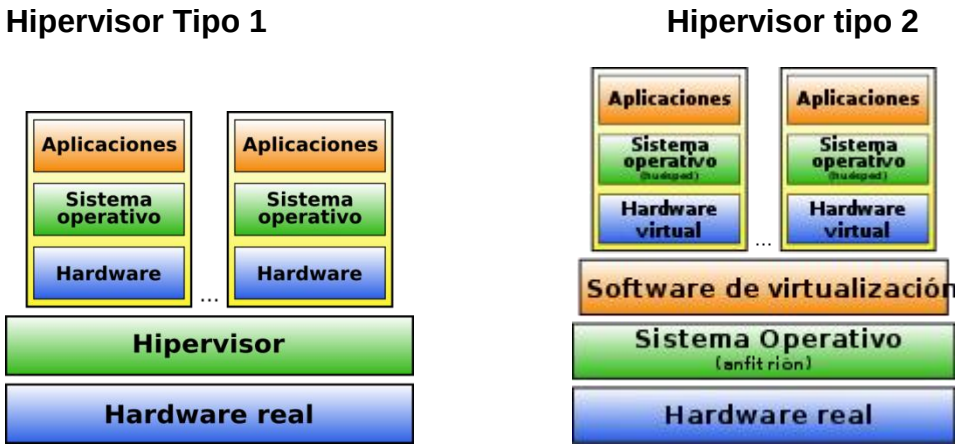



*Figura 6. Tipos de Hipervisor. Fuente: sliceoflinux.wordpress.com*

### 3.2.2. Ventajas y desventajas

Algunas ventajas importantes de la virtualización son las siguientes:

- Facilidad de administración y operación.
- Facilidad de generar instancias virtuales.
- Permite clonación de imágenes.
- Flexibilidad para asignar una mayor cantidad de recursos computacionales (Memoria, Disco, Procesador e Interfaces de red).
- Reducción de procesos administrativos con la homogeneización de máquinas.
- Aislamiento de cada una de las máquinas virtuales que les permite no afectar a las demás en caso de fallo.
- La seguridad que presenta cada máquina no compromete al sistema operativo huésped.
- La portabilidad, ya que una máquina puede ser migrada fácilmente de un servidor a otro.

La principal desventaja, es la necesidad de equipos con características de hardware robustas, esto debido a la pérdida de rendimiento a mayor cantidad de

máquinas virtuales instanciadas, fundamentalmente por saturación de operaciones de entrada y salida.

## 3.3. Computación en la Nube

Este paradigma ha tenido varias definiciones desde la aparición del concepto hasta la realización de este proyecto [4], [5], [7], [8], [11], algunas plantean enfoques históricos, comerciales y otras un enfoque tecnológico, pero aún no han llegado a un común acuerdo. Sin embargo para efectos de presentación de este proyecto se presentará la definición, características esenciales y modelos de servicio planteados por el Instituto Nacional de Estándares y Tecnología de los Estados Unidos, NIST por sus siglas en inglés.

### 3.3.1. Definición

*"Es un modelo que permite el acceso por red a un conjunto compartido de recursos computacionales configurables (ej. Redes, servidores, almacenamiento, aplicaciones y servicios) de manera ubicua, conveniente y bajo demanda, que pueden ser entregados y proporcionados rápidamente, con un mínimo de esfuerzo administrativo o de interacción del proveedor de servicios."* [8]

### 3.3.2. Características esenciales

*Auto servicio bajo demanda.* Un usuario puede unilateralmente aprovisionarse de capacidades de cómputo, tales como uso de servidor, almacenamiento en red, la medida que necesite, de forma automática y sin requerir de la interacción humana del proveedor de servicio.

*Amplio acceso por red.* Las capacidades de cómputo están disponible en la red y son accesibles a través de mecanismos comunes que permitan el uso de plataformas heterogéneas para los usuarios (ej. teléfonos móviles, tablets, computadores portátiles y de escritorio).

*Recursos comunes compartidos.* Los recursos de cómputo del proveedor son comunitarios y permiten servir a múltiples usuarios usando un modelo de infraestructura compartida, con diferentes recursos físicos y virtuales asignados dinámicamente y reasignados de acuerdo a la demanda de los usuarios. Existe la sensación de independencia de la ubicación en la cual el usuario no tiene control o conocimiento acerca de la ubicación exacta de los recursos aprovisionados, pero puede existir la posibilidad de especificar la ubicación en un nivel más alto de abstracción (ej. país, estado o centro de cómputo). Ejemplos de recursos incluyen almacenamiento, procesamiento, memoria y ancho de banda.

*Elasticidad rápida.* Las capacidades de cómputo pueden ser aprovisionadas y liberadas rápidamente y en algunos casos de forma automática para escalar rápidamente aumentando o disminuyendo la cantidad para hacer frente a la demanda. Para el usuario, las capacidades disponibles para aprovisionamiento pueden parecer ilimitadas y puede adquirir cualquier cantidad en cualquier momento.

*Servicio medible.* Los sistemas de computación en la nube controlan y optimizan automáticamente el uso de recursos suministrando una capacidad de medición bajo algún nivel de abstracción apropiado al tipo de servicio (ej. almacenamiento, procesamiento, ancho de banda y usuarios concurrentes). El uso de recursos puede ser monitoreado y reportado, permitiendo un servicio transparente para el proveedor y para el consumidor del servicio utilizado.

### 3.3.3. Modelos de servicio

*Software as a Service (SaaS)*. La capacidad de cómputo proporcionada al usuario, es el uso de las aplicaciones del proveedor que se encuentran en ejecución en la infraestructura cloud. Las aplicaciones son accesibles desde varios dispositivos de usuario a través de interfaces sencillas, como un navegador web (ej. web email), o una aplicación de escritorio. El usuario no administra o controla la infraestructura subyacente sobre la cual se ejecuta, como hardware, servidores, sistemas operativos, almacenamiento o capacidades de control sobre la aplicación; con la posible excepción de preferencias de configuración de la aplicación específicas para el usuario.

*Platform as a Service (PaaS).* La capacidad de cómputo proporcionada al usuario es la de desplegar aplicaciones adquiridas o creadas por el usuario, usando lenguajes, librerías, servicios y herramientas soportadas por el proveedor. El usuario no administra o controla la infraestructura subyacente sobre la cual se ejecuta, incluyendo redes, servidores, sistemas operativos o almacenamiento, pero tiene control sobre las aplicaciones desplegadas y posiblemente sobre las preferencias de configuración del entorno de ejecución de la aplicación.

*Infrastructure as a Service (IaaS).* La capacidad de cómputo proporcionada al usuario es el aprovisionamiento de redes, procesamiento, almacenamiento y otros recursos computacionales fundamentales sobre los cuales el cliente puede desplegar y ejecutar aplicaciones según su criterio, lo cual incluye sistemas operativos y aplicaciones. El usuario no administra o controla la infraestructura subyacente pero tiene control sobre los sistemas operativos, almacenamiento y aplicaciones desplegadas; y posiblemente un control limitado sobre algunos componentes de red seleccionados (ej. firewall).

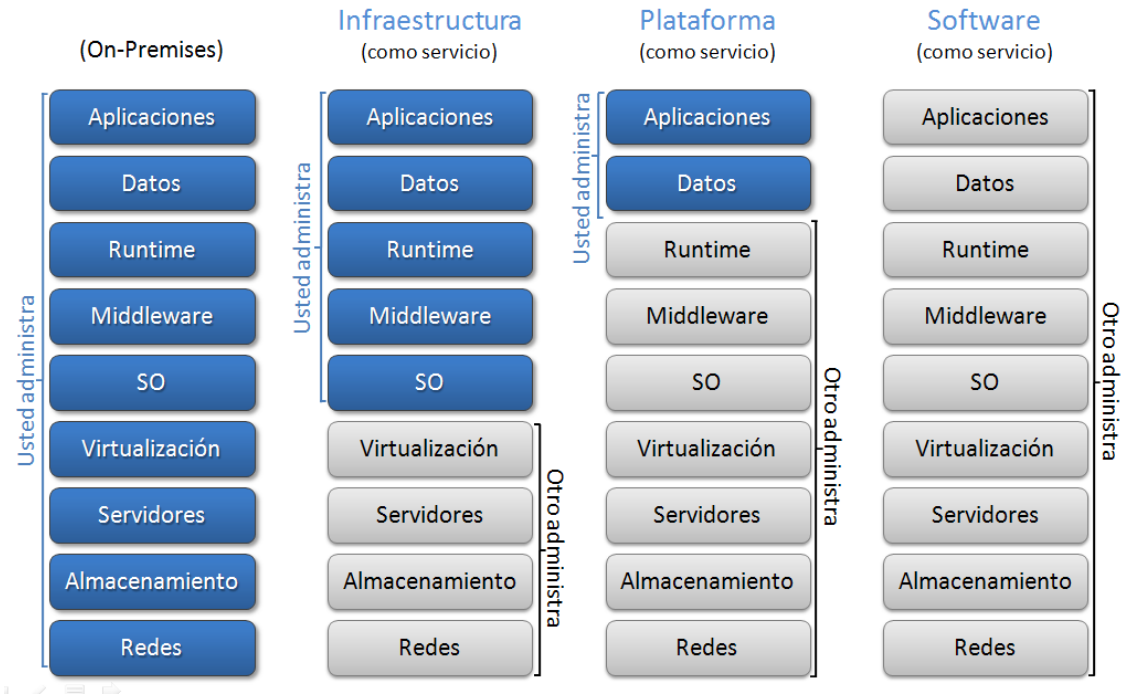



*Figura 7. Modelos de Servicio. Fuente: Carreño Díaz, Emmanuel. Modelo y prototipo de servicios de computación en la nube para estudiantes y profesores de la escuela de ingeniería de Sistemas e informática de la universidad industrial de Santander. Bucaramanga. Universidad industrial de Santander. Facultad de Ingenierías Fisicomecánicas. Escuela de ingeniería de sistemas e informática, 2012. 25 p. [7]*

En la Figura 7 se puede observar una comparación directa entre los diferentes tipos de modelos de servicios de computación en la nube y se indican los niveles de administración permitida al cliente y los que son responsabilidad del proveedor del servicio, junto al modelo tradicional servicio. *On-premises* quiere decir que el servicio de cómputo es alojado en las instalaciones del usuario, por lo tanto este administra la totalidad de los componentes, algo que puede ser innecesario según el tipo de empresa, la administración de redes, almacenamiento, servidores, hipervisores, sistemas operativos, *middleware* (software intermediario), *runtimes* (entornos de ejecución), aplicaciones y datos pueden ser manejados por un tercero según el modelo seleccionado.

# 4. ANÁLISIS Y DISEÑO

## 4.1. Estudio de la situación actual

La infraestructura de servicios del grupo CONUSS está construida en base a una capa básica de hardware conformada por equipos de tipo servidor de altas prestaciones. Estos equipos fueron adquiridos gracias a una donación de la fundación Raúl D. Ocazionez y permitieron la creación del esquema inicial, trabajo surgido en base a un proyecto académico [1], que se encargó de plantear la infraestructura para generar un modelo de servicios con las siguientes características principales:

- Sistemas basados en distribuciones Linux.
- Servidores (web, base de datos, archivos, SMTP) y software especializado para soportar el hospedaje de aplicaciones web.
- Prototipo de modelo de alta disponibilidad para garantizar la prestación constante de los servicios.
- Políticas para la administración, monitoreo y prestación de servicios.

Más adelante, con la ejecución y culminación de un nuevo proyecto [3], se plantea un prototipo de computación en la nube basado en el esquema de prestación de servicios IaaS (Infraestructure as a Service) con el fin de apoyar la investigación y ejecución de proyectos de grado, mediante la creación de plataformas virtuales con ambientes simulados. Este esquema utiliza la misma infraestructura inicial, aunque aborda un enfoque diferente. Producto de este proyecto surgió una plataforma denominada CloudEISI que planteó un sistema funcional con las siguientes características:

- Servicio de asignación de instancias virtuales[8] con software personalizado según el proyecto o el área de investigación del usuario.
- Repositorio de discos duros virtuales con servicios preconfigurados que hacen las veces de plantillas para las plataformas virtuales.
- Un esquema de red interna basado en interfaces virtuales para la creación de redes virtuales entre las instancias.
- Una interfaz administrativa para el manejo y asignación de recursos virtuales.
- Políticas de prestación de servicios a los usuarios.
- Un punto de acceso de red para la comunicación de los servicios virtuales con el mundo exterior.

Los sistemas anteriormente descritos fueron puestos a prueba observando su comportamiento durante el periodo de prestación de servicio. Gracias a la documentación administrativa y a los manuales de uso de los principales componentes de software instalados dentro de los equipos, fue posible abstraer las principales configuraciones de los sistemas base y de esa forma hacer una comparación a modo de evaluación entre las configuraciones actuales y un estado ideal de funcionamiento, creado a partir de las buenas prácticas y recomendaciones consignadas en recursos bibliográficos como normas y guías de seguridad, manuales de administración, guías de funcionamiento de paquetes de software y ejemplos documentados de implementación en otros contextos.

Además de lo anterior y como punto principal abordado inicialmente dentro de los objetivos de este proyecto, el grupo tiene interés en ampliar la infraestructura, incorporando nuevos nodos y equipos de apoyo al clúster. Tomando en cuenta

[8] Se define instancia virtual, a la creación de una máquina virtual en base a la copia de una imagen o disco virtual, que actúa como una plantilla con un sistema operativo base y paquetes de software incorporados en base al propósito de la máquina.

esto, se determinó otro punto de evaluación basado en propiedades como la escalabilidad.

A continuación se expondrá el análisis de cada modelo, partiendo del modelo general y luego a los modelos en cuestión. En el proceso se detallará la forma de funcionamiento de cada uno, así como un análisis comparativo que permita establecer las carencias y dificultades presentadas en base a los criterios seleccionados con anterioridad. Esto establecerá una base para la deducción de los requerimientos sobre los cuales se creará la propuesta del nuevo modelo.

## 4.2. Esquema general de la infraestructura

Abstrayendo la configuración física de red del entorno, se crea un esquema representado por la siguiente figura:

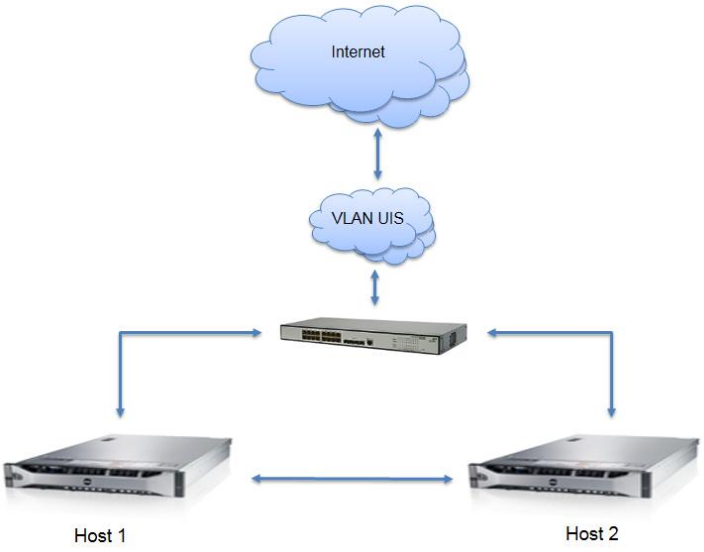



*Figura 8. Esquema General de la Infraestructura. Fuente: Los Autores*

Como se puede ver en la figura, los hosts están conectados a la red universitaria que brinda el servicio de conexión a la red de internet. Cada uno de los host tiene su propia conexión, siendo este el punto de salida de los servicios que alojan respectivamente. Los usuarios se valen tanto de la red universitaria como de la red

externa para acceder a los servicios, siendo este el enfoque planteado desde el proceso de implementación de la infraestructura.

Existe una red interna dedicada entre los hosts, dicha red apoya los sistema de HA y Cloud Computing quienes debido a sus exigencias de comunicación, necesitan una red libre de tráfico y con velocidad y latencia adecuadas.

### 4.3. Modelo de computación en la nube

Uno de los sistemas que se vale del esquema anteriormente descrito es la plataforma de computación en la nube conocida como CloudEISI. Tal como se mencionó en una sección anterior, el sistema se basa en el modelo de prestación de servicios IaaS (Infraestructures as a Service) administrando recursos virtuales que los usuarios pueden utilizar para crear entornos personalizados en base a los requerimientos de desarrollo o investigación solicitados. A continuación se expone el modelo general del sistema.

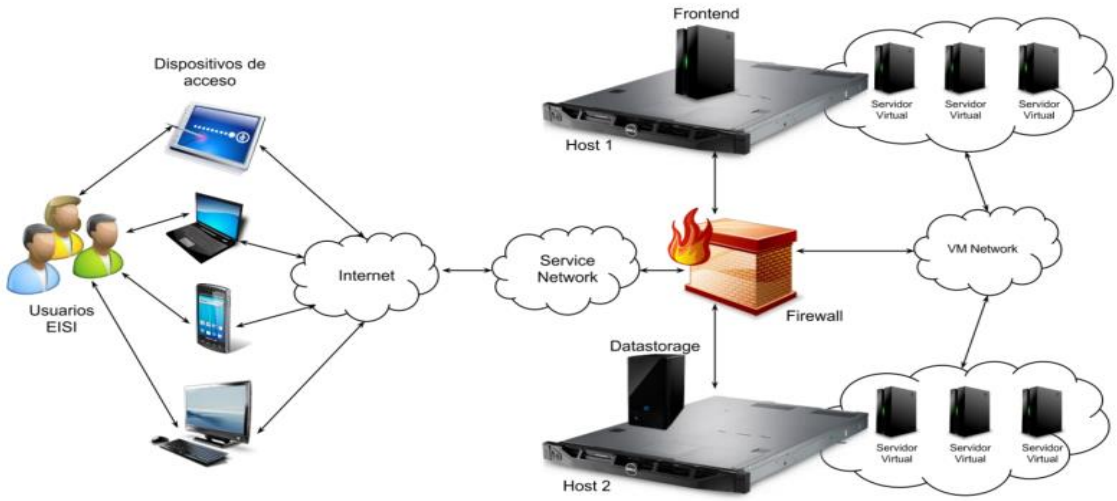



*Figura 9. Modelo de Prestación de Servicios para la EISI. Fuente: Carreño Díaz, Emmanuel. Modelo y prototipo de servicios de computación en la nube para estudiantes y profesores de la escuela de ingeniería de Sistemas e informática de la universidad industrial de Santander. Bucaramanga. Universidad industrial de Santander. Facultad de Ingenierías Fisicomecánicas. Escuela de ingeniería de sistemas e informática, 2012. 47 p. [3]*

El modelo cuenta con 6 elementos principales:

**Frontend**: Es el servidor principal encargado de administrar la infraestructura completa, controlando recursos como las instancias y redes virtuales, los datastores y los hosts físicos. También desempeña tareas como el filtro y redirección de los paquetes desde la red externa (Internet, Service network) hacia la red interna (VM Network) y la asignación de instancias a los host en base a una política de gestión de recursos.

**Datastore**: Almacena las imágenes virtuales que serán usadas como plantillas en la asignación de instancias.

**Servidor virtual (instancia virtual)**: Son máquinas virtuales instanciadas a partir de las imágenes del datastore y en ejecución dentro de alguno de los hosts. El frontend mediante su sistema de asignación de recursos, realiza una copia de la imagen base hacia el host elegido, estableciendo dentro de este las características de la máquina virtual a crear para luego ponerla en ejecución.

**Host**: Equipo físico reconocido como “nodo de instanciación” por el frontend. Almacena y ejecuta las instancias virtuales asignadas por este último.

**VM Network:** Red interna de la infraestructura para el envío y recepción de paquetes entre las instancias virtuales y el frontend.

**Service Network:** Red que provee el punto de acceso a la red externa. Utilizada por el frontend para el envío y recepción de datos desde y hacia las instancias virtuales.

### 4.3.1. Análisis general del modelo

Una de las ventajas del sistema es su arquitectura modular. Las instancias virtuales pueden ser consideradas como paquetes funcionales con procesos personalizados de acuerdo a la función con la cual fueron creadas, independientes

de otras instancias, capaces de ser migradas[9] entre un nodo u otro sin perder la esencia de su capa funcional (interfaz de usuario) y manteniendo la consistencia de los datos almacenados en ellos. Esta es una propiedad heredada de los sistemas virtualizados. Además, frente a un fallo en uno de los servicios o instancias, no se ve involucrado el sistema en general.

A pesar de esto y por la forma en que está construido el sistema, no es posible asegurar un porcentaje de confiabilidad alto frente al esquema de servicios planteado, el cual demanda funcionamiento continuo y con periodos de corte del servicio relativamente pequeños. Esta exigencia surge implícitamente al considerar el tipo de aplicaciones y datos manejados por los usuarios. Aunque la mayoría de instancias virtuales son utilizadas como entornos de prueba que no albergan datos críticos o sensibles, por otra parte se destinan instancias especiales para el hospedaje de sitios y aplicaciones que funcionan en un entorno de producción y por tanto demandan características del sistema como la seguridad, confiabilidad y disponibilidad.

Los problemas en cuestiones como la confiabilidad y la disponibilidad surgen cuando se toman en cuenta escenarios como el de la caída de uno o más nodos. De acuerdo a las limitaciones de disposición de equipos servidores en el proceso de implementación, detalladas en el documento del proyecto original [3], los servicios del frontend fueron alojados dentro de una máquina virtual para emular las características exigidas por la plataforma de administración[10]. De esta forma, uno de los nodos cumple una doble función al almacenar esta parte del sistema y a su vez las instancias virtuales que ella genera. Al ser el frontend el único punto de acceso a la red externa desde la perspectiva de las instancias virtuales, fácilmente se pueden deducir las consecuencias ante el evento de la caída del nodo en cuestión. El datastore no tiene tal nivel de prioridad, ya que solo es

[9] En los sistemas virtualizados, la migración es el traslado de los recursos virtuales de un nodo a otro conservando las características iniciales mediante el respaldo del estado actual.
[10] Al usar este término, implícitamente se hace alusión a la herramienta OpenNebula, software especializado para la administración de esquemas de Cloud Computing usado en el modelo actual.

necesario durante procesos como la creación, almacenamiento y borrado de instancias virtuales, pero de la misma forma retrasaría la puesta en marcha de los procesos en cuestión.

La plataforma de administración permite anticipar caídas o cortes programados por mantenimiento de la infraestructura, brindando soluciones como la migración o respaldo de los servicios, pero esto no funciona de la misma manera en situaciones que no hacen parte de una planeación previa.

El nuevo nodo que se va a incorporar pasaría a soportar el proceso de asignación de recursos virtuales. Una de las ventajas es el aprovechamiento de la capacidad extra de hardware ya sea para reducir la carga de recursos físicos utilizados en los otros nodos o para incrementar la capacidad de asignación y ejecución de servicios virtuales.

En base al análisis hecho sobre los sistemas en producción, dicho proceso está soportado y documentado por el software administrador de la plataforma. En cuanto al nivel de complejidad y el impacto sobre los servicios en funcionamiento, no se presentan incidencias serías debido a la claridad y limpieza del proceso que involucra ciertos cambios sobre el frontend y no sobre los nodos actuales.

### 4.4. Modelo de alta disponibilidad

El sistema de alta disponibilidad permite garantizar el funcionamiento constante de los servicios alojados dentro de la infraestructura a través procesos de monitoreo y control. El sistema anticipa fallas y caídas tanto a nivel de host como a nivel de recursos (servicios). Se vale de una heurística especializada que le permite determinar el estado más adecuado del clúster, calculado y decidido en respuesta a ciertos eventos dentro de la infraestructura.

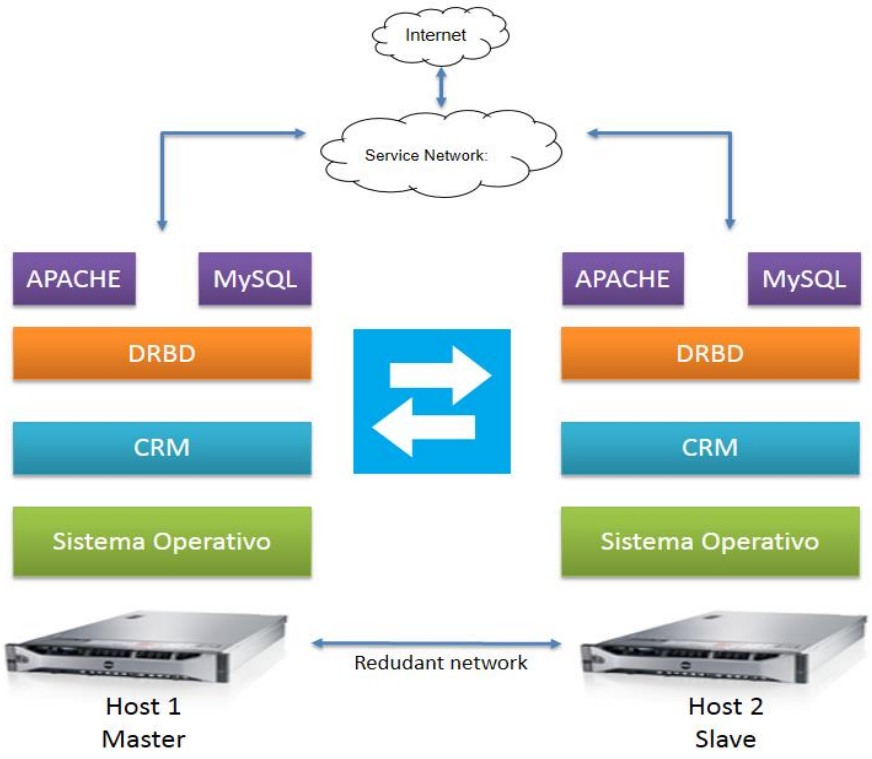



*Figura 10. Modelo de Alta Disponibilidad Anterior. Fuente: Los autores*

El esquema en producción puede observarse en la figura y comprende los siguientes componentes principales:

**Sistema Operativo:** Sistema base sobre el que se instalan y ejecutan los programas y paquetes de software que soportan los servicios de la plataforma.

**CRM:** Por sus siglas en inglés, Cluster Resource Manager, es un middleware que ofrece una capa sobre la cual se sitúan los servicios que van a hacer parte del sistema de alta disponibilidad. Su función es monitorear, controlar y migrar los recursos que tiene a cargo con el fin de garantizar su funcionamiento ininterrumpido y aplicar acciones correctivas cuando se presenta un comportamiento anormal.

**DRDB:** Es un sistema de redundancia capaz de respaldar el contenido completo de un bloque de datos en tiempo real, similar al comportamiento de los sistemas

de RAID incorporados en muchos dispositivos. Mantiene un respaldo de datos sensibles como sitios web y base de datos.

**APACHE:** Servidor HTTP de código abierto encargado de recibir las peticiones de los usuarios y responder con el recurso web adecuado.

**MySQL:** Sistema gestor de base de datos que almacena los datos utilizados por los sitios web y aplicaciones alojados en el servidor.

### 4.4.1. Análisis general del modelo

El sistema de alta disponibilidad, a diferencia del sistema de Computación en la Nube, reposa directamente sobre la capa física de la infraestructura, es decir que sus componentes base, mencionados en la sección anterior, están implementados directamente sobre el sistema operativo y utilizan las interfaces de red físicas de los equipos sin recurrir a algún tipo de virtualización[11].

El sistema funciona mediante una política de roles clásicamente conocida como Maestro/Esclavo, esto quiere decir que solo uno de los pares (nodos con características similares) está habilitado para correr los servicios y atender a las peticiones de los usuarios, mientras que el miembro restante del clúster se mantiene en modo de espera, vigilante de la actividad de su semejante, copiando a modo de espejo los cambios en los datos y tomando el control (rol de maestro) cuando se detecta una falla en el sistema principal.

Según este modo de funcionamiento se plantean varias cuestiones: A diferencia del sistema tratado en una sección anterior, los servicios están implementados directamente sobre la capa de hardware de los equipos miembros del clúster. Esto por un lado tiene como ventaja el aprovechamiento completo de toda la capacidad

[11] Tipos de virtualización. Kusnetzky, Dan. “Virtualization: A Manager’s Guide”.2011. Estados Unidos. 2 p.

de hardware por parte de los servicios, pero también representa cierto desperdicio si las aplicaciones no ejecutan procesos complejos o pesados. Esto deja un margen de capacidad ociosa realmente grande, factor que conlleva a ciertos costos operativos en especial costos energéticos[12]. Al estar uno de los nodos inoperativo en sus servicios, es evidente el escenario descrito anteriormente.

Tomando en cuenta factores como la escalabilidad y el alojamiento de servicios especiales, el sistema es flexible hasta cierto punto. El sistema de control CRM cuenta con opciones especiales que le permiten incorporar nuevos hosts al clúster permitiendo gestionar recursos en alta disponibilidad a un tercer nivel, creando de esta manera esquemas más complejos tal como los mencionados en la sección teórica de este documento. El problema surge al intentar extender el sistema de replicación a los nuevos nodos del clúster. El sistema, tal como se detalla en el modelo presentado en la figura, es utilizado en entornos que no cuentan con un área de almacenamiento dedicada, ofreciendo confiabilidad, que es una característica de dispositivos especiales de almacenamiento como las SAN y NAS. DRBD solo soporta 2 modos de operación: Activo/pasivo y Activo/Activo implementados mediante 2 equipos idénticos. El primero es un modo compatible con la política de maestro/esclavo. El segundo se utiliza en infraestructuras que quieran aprovechar todos sus recursos de hardware simultáneamente, aplicando un balanceo de carga en el acceso a los datos. Para esto, las aplicaciones alojadas en el clúster deben soportar el acceso concurrente a los datos y no ejercer controles estrictos sobre ellos. Debido a las medidas de seguridad y control ejercidas por gestores de bases de datos como MySQL para controlar las propiedades de una transacción, no es posible mediante este sistema mantener dos instancias de este paquete software ejecutándose de forma simultánea.

[12] Basado en el análisis del proyecto de Cloud Computing [3] incial.

El sistema de replicación no soporta esquemas de más de dos nodos a menos de que actúe como área de almacenamiento dedicada. Se presenta entonces una ruptura en el enfoque de nodos idénticos del clúster y es necesario la utilización de paquetes de software especializados para solventar dicho inconveniente añadiendo en el proceso una capa de complejidad extra al modelo.

## 4.5. Consideraciones iniciales de diseño

En la creación del nuevo modelo de la infraestructura, se realizó un análisis sobre la función que debería tener el nuevo nodo tomando en cuenta los modelos presentes. Partiendo de las necesidades expuestas de forma general en la sección anterior y de aquellas encontradas en un análisis más exhaustivo de los componentes funcionales de cada modelo, se inició una fase de investigación en la que se examinaron diferentes esquemas de infraestructuras de IT, así como métodos de diseño extraídos de otros enfoques de la ingeniería de sistemas como por ejemplo la ingeniería del software, que en esencia establece pautas y principios que tienen cierta aplicabilidad en el área.

Con base en la investigación realizada, el análisis de los modelos presentes y la extensión de la infraestructura mediante la incorporación de un nuevo host, se determinó que el nuevo modelo debe contar con las siguientes características:

- Un diseño modular, es decir que las aplicaciones y servicios no deben estar ligadas al host que las hospeda, a menos que tenga restricciones especiales que la aten a la infraestructura.
- Redundancia de servicios y datos sensibles.
- Un balanceador de carga para distribuir entre los host el uso de los recursos.
- Establecer un área de almacenamiento dedicada para las imágenes usadas por la herramienta de IaaS.
- Virtualización de los servicios.

- Alta disponibilidad del punto de acceso a la red de las instancias virtuales que en esencia se ve representado por el frontend.
- Capacidad de migrar del modelo base al nuevo modelo sin realizar interrupciones significativas sobre los servicios en producción.

### 4.6. Diseño del modelo

El diseño parte desde la configuración base de los sistemas. Los servidores por defecto contienen el sistema operativo necesario para la implementación de los demás componentes. Las capa base, estará conformada por el hardware, el sistema operativo con algunos paquetes de software de administración y el hipervisor que administra y controla las instancias virtuales. Al hacer esto se aísla y divide la funcionalidad en pequeños componentes. Cada servicio, separado desde la perspectiva de su funcionalidad, será incorporado dentro de una máquina virtual. Este enfoque es tomado de uno de los principios de desarrollo de software conocido como diseño basado en componentes [9].


*Figura 11. Esquema base de diseño. Fuente: Los autores*

Tomando en cuenta este principio, si se puede dividir el modelo en unidades lógicas y funcionales con una interfaz definida, es posible obtener un diseño simple y fácilmente administrable. Los servicios no están restringidos por el sistema operativo y pueden ser configurados, modificados, y migrados sin afectar las demás partes.

Siguiendo este principio, el siguiente paso fue integrar los modelos de alta disponibilidad y computación en la nube estudiados con anterioridad en un solo modelo.

### 4.6.1. Diseño del modelo de alta disponibilidad

En el nuevo enfoque de servicios, el modelo de alta disponibilidad es modificado con el fin de brindar respaldo a instancias virtuales sensibles. A diferencia del modelo antiguo, no solo se estableció una redundancia a nivel de datos, también se buscó mantener una replicación a nivel de configuraciones y en general a nivel de componentes es decir asegurando un respaldo de toda una instancia virtual.

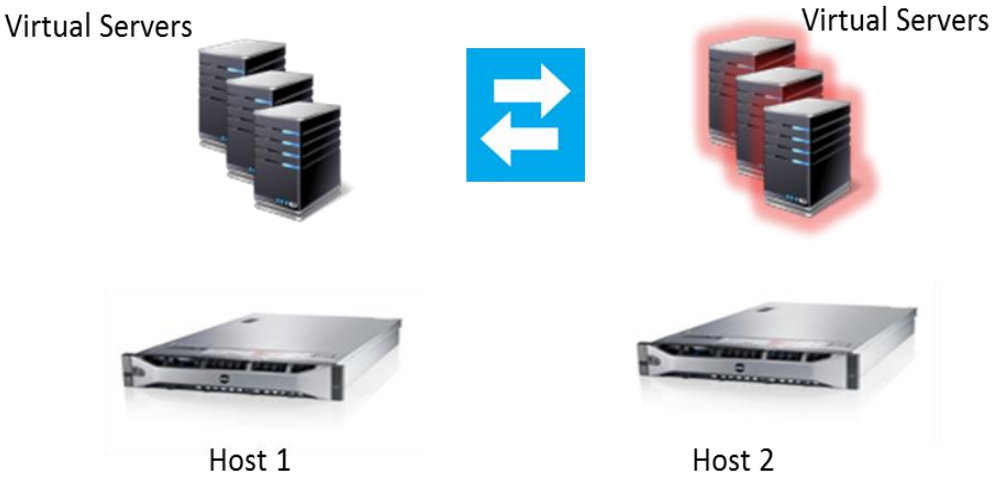



*Figura 12. Diseño del Modelo de Alta Disponibilidad. Fuente: Los autores*

### 4.6.2. Diseño del modelo de computación en la nube

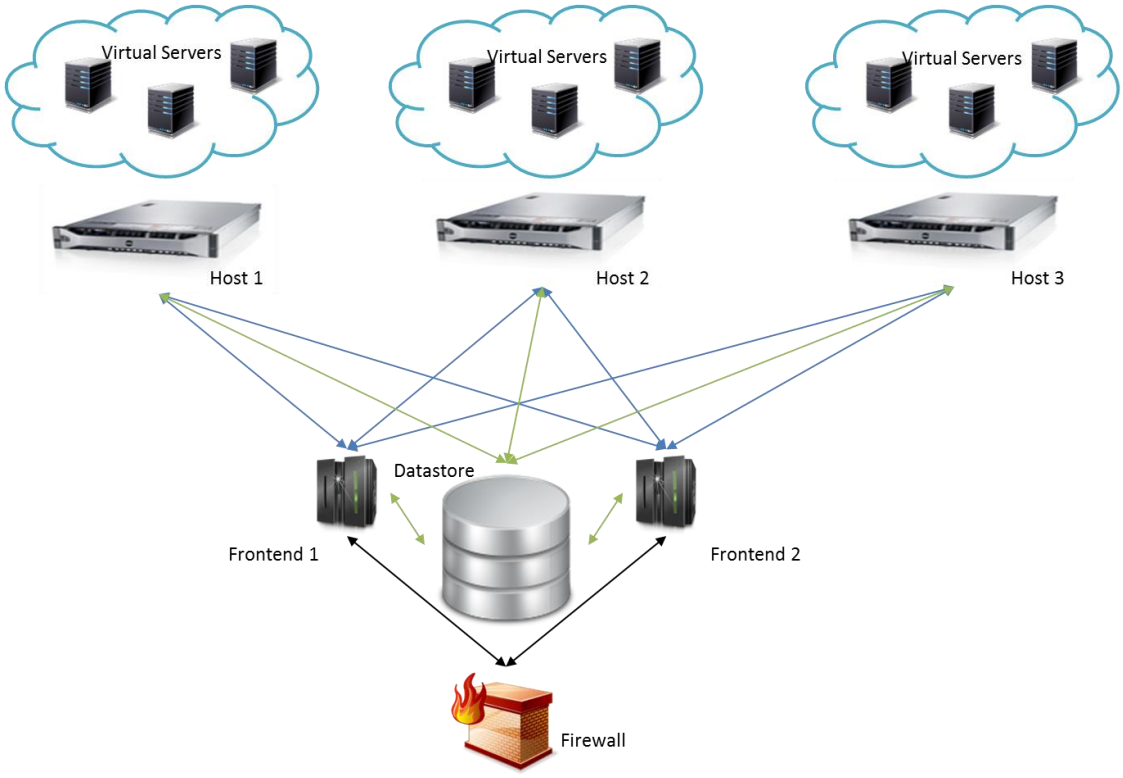



*Figura 13. Diseño del Modelo de computación en la nube. Fuente: Los autores*

El esquema sigue un arreglo clásico de infraestructuras de IT conocido como modelo 3-2-1 o jerárquico [2]. En él se tienen 3 hosts que hospedan las instancias virtuales, dedicando toda su capacidad de hardware a ello. Luego existe un servidor redundante que administra la asignación de instancias y a la vez desempeña el rol de punto de acceso a la red. Por último, un área de almacenamiento de imágenes utilizadas como plantillas para la creación de instancias virtuales.

Una de las ventajas de este esquema es su gran capacidad de escalabilidad, esto debido a la separación de componentes.

### 4.6.3. Modelo de soporte para los sistemas en producción

Los modelos creados tienen como fin consolidar el entorno en producción, agregando nuevas características que optimicen y aseguren modelos existentes. Pensando en la administración y soporte, se diseñó un entorno alterno que sirviera como base para la validación de los componentes en producción así como la creación de nuevos componentes con una funcionalidad asignada. Esto brinda al administrador una herramienta para aplicar cambios al sistema sin recurrir a acciones empíricas sobre el mismo entorno, reduciendo de esta forma la probabilidad de falla.

Este modelo fue construido considerando las capacidades extras de hardware en el grupo CONUSS y al analizar la mejor manera de integrarlo a las actividades de investigación del grupo.

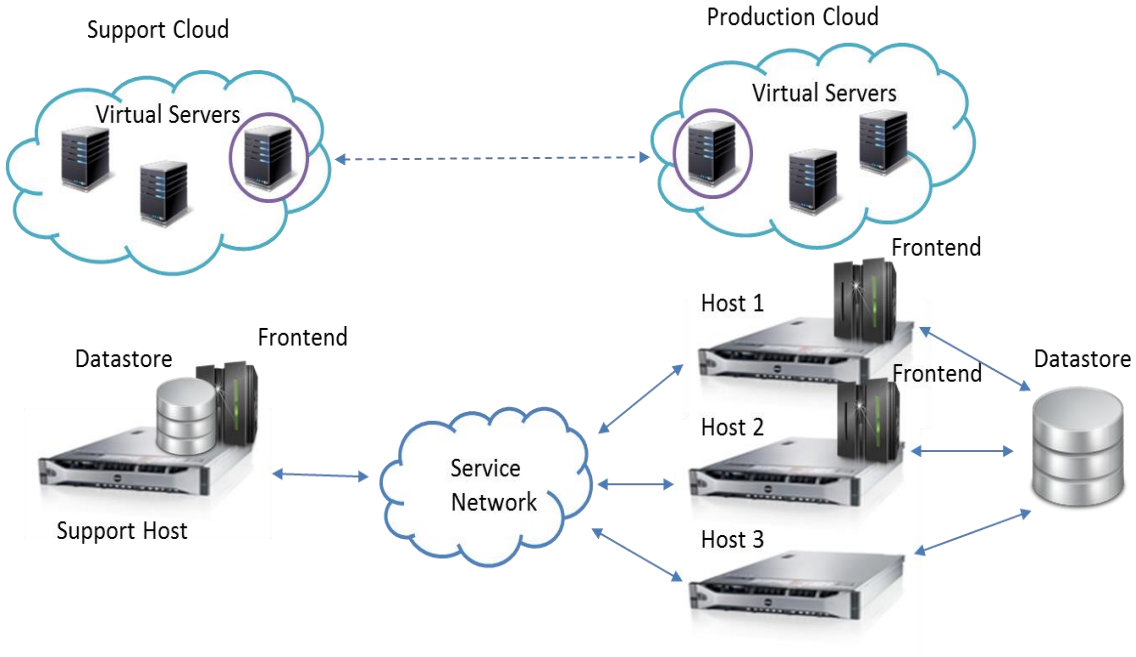



*Figura 14. Modelo de soporte para los sistemas en producción. Fuente: Los autores*

En el entorno de pruebas, se asigna un servidor de instancias y un repositorio para el almacenamiento de imágenes. Estos poseen características similares a sus homólogos en el entorno en producción, de esta manera se obtiene un acercamiento entre las dos infraestructuras permitiendo el traspaso limpio de componentes entre ellas.

### 4.6.4. Modelo de balanceo y distribución de recursos

Con el enfoque modular de la infraestructura y la capacidad extra de hardware obtenida con la adquisición de un nuevo equipo servidor, se busca hacer una distribución adecuada de los recursos, separando la carga que ejercen los servicios en funcionamiento sobre cada uno de los equipos. Un esquema muy utilizado en las infraestructuras actuales se basa en la redundancia de servicios que pueden actuar de forma simultánea soportados por un sistema de balanceo de carga. Dicho sistema evita la saturación excesiva de trabajo sobre un solo nodo a causa de una concurrencia elevada de usuarios en el sistema.

En el esquema actual, este tipo de sistemas puede no solo distribuir la carga entre varios servicios redundantes, sino que también agrega una capa extra que brinda alta disponibilidad a las instancias virtuales cobijadas y una capa de seguridad en los accesos, que serían transparentes para el usuario debido a que no se expone la estructura de la arquitectura y en caso de fallas se recupera de forma oportuna el servicio.

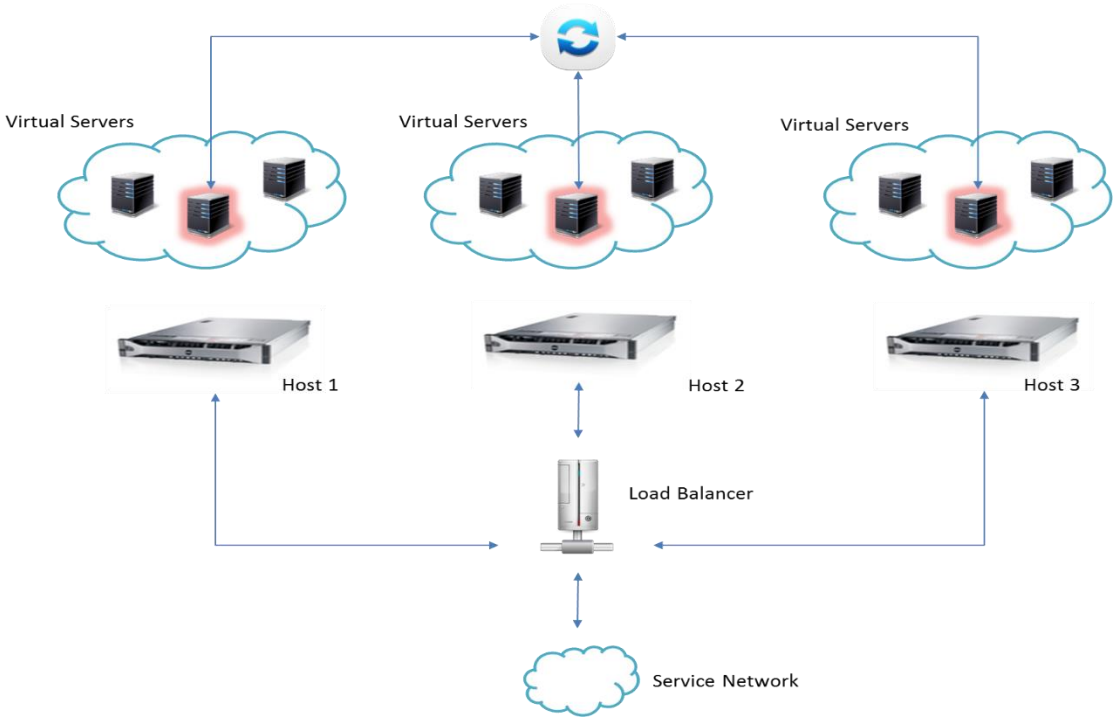



*Figura 15. Diseño del Modelo de balanceo y distribución de recursos. Fuente: Los autores*

El esquema en la figura 15, parte de los modelos descritos anteriormente pero incorpora dos nuevas características: Un balanceador de carga (Load Balancer) que inteligente distribuye las peticiones entre los servicios que tiene a cargo y un sistema de sincronización, capaz de transmitir los cambios hechos en una de las instancias virtuales hacia las demás instancias en tiempo real manteniendo la consistencia y disponibilidad de los datos.

# 5. IMPLEMENTACIÓN DEL MODELO

Durante el proceso de implementación del modelo, se hizo un análisis de las herramientas necesarias para dar soporte a un servicio de alta disponibilidad de la infraestructura de acuerdo a los requerimientos planteados en la fase de análisis y diseño del modelo y además de ello debían cumplir con unos requisitos planteados desde el inicio del proyecto.

- Las herramientas usadas deben ser gratuitas.
- No debe existir incompatibilidad de licencias o limitaciones de uso.
- Deben funcionar sobre Debian (sistema operativo de los servidores).

Selección del hipervisor

En el análisis de la selección se decidió mantener el esquema de la infraestructura bajo el hipervisor Virtualbox. Esto se hizo pensando en la continuidad de los servicios que hasta el momento funcionaban en el sistema y en la relativa facilidad de hacer la migración y modificación de los módulos que cobija el nuevo modelo. También se analizaron hipervisores como KVM y XEN, tecnologías desarrolladas con un gran soporte para entornos Linux, que por sus prestaciones y características pueden agregar nuevas funcionalidades y mejoras a la infraestructura. El problema radica en los requisitos esquemáticos y de software que requieren, que para un entorno que actualmente se encuentra en producción podría suponer un tiempo de implementación mucho más largo y con periodos de inactividad que serían críticos para la prestación del servicio.

## 5.1. Herramienta de administración para IaaS

Para el desarrollo de este proyecto se actualizó la versión del OpenNebula que funcionaba anteriormente en la plataforma 3.2.1 a la versión más reciente y estable 3.8.1. Esta versión incluye nuevas mejoras tanto en seguridad como en funcionamiento y plantea un nuevo enfoque que hace más fácil y estable el proceso de implementación. Quizás una de las características a resaltar es su compatibilidad con infraestructuras con múltiples sistemas de virtualización, propiedad que es de vital importancia en futuras investigaciones realizadas por el grupo.

En la siguiente figura se detalla la interfaz principal del OpenNebula versión 3.8.1.

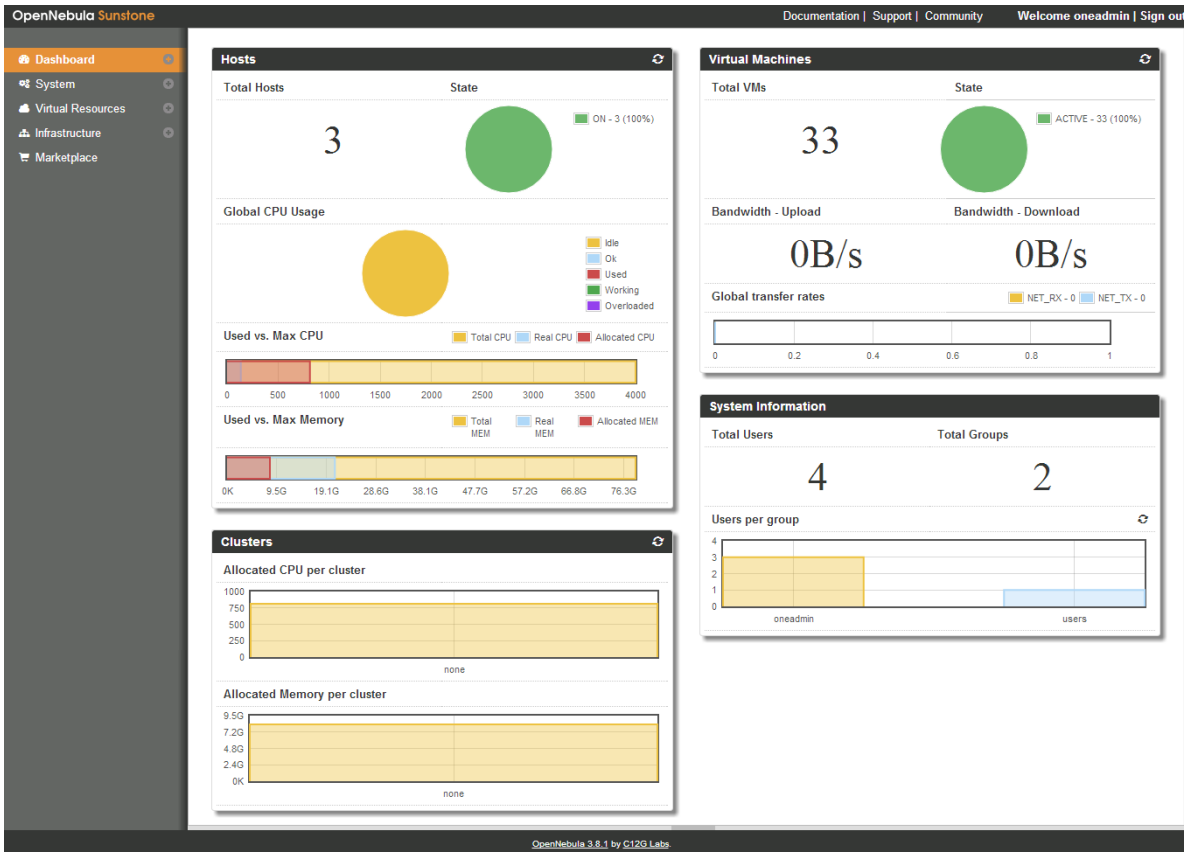



*Figura 16. Interfaz de administración de OpenNebula. Fuente: Los autores*

## 5.2. Herramientas de alta disponibilidad

Para la creación del clúster de alta disponibilidad se utilizaron las herramientas de software libre DRBD, Pacemaker, Corosync/OpenAIS, y se decidió usar la infraestructura de Clúster Activo/Pasivo, tal como se expuso en la fase de diseño.

Antes de la instalación de las herramientas nombradas se llevó a cabo un proceso de investigación determinando las herramientas más convenientes para su utilización, partiendo de varias alternativas de software, se optó por elegir las mencionadas por distintos factores:

- Todas las herramientas software son de carácter libre.
- Tienen documentación completa y soporte actualizado.
- Buenas características de seguridad.
- Adaptables a distintos esquemas de funcionamiento según sea requerido.
- Se complementan muy bien como conjunto, brindando una infraestructura de clúster muy eficiente.

### 5.2.1. DRBD (Distributed Replicated Block Device)


*Figura 17. Logotipo DRBD. Fuente: http://www.drbd.org/*

Es un software que permite la creación de sistemas de almacenamiento distribuido y opera en sistemas GNU/Linux. Esto se logra mediante la duplicación de un dispositivo (disco duro, partición, volumen lógico, etc.) a través de una red

asignada. DRBD puede entenderse como una red basa en RAID-1 (copia exacta o espejo de un conjunto de datos en dos o más discos).

Todos los recursos en DRBD tiene un rol que puede ser primario o secundario.

- Un dispositivo DRBD con rol primario se puede utilizar sin restricciones de lectura y escritura. Se puede usar para crear sistemas de archivos sin formato, E/S directa al dispositivo de bloques, ext3, ext4, volúmenes lógicos, entre otros.
- Un dispositivo DRBD con rol secundario recibe las actualizaciones de un dispositivo del mismo nivel, pero contrariamente no permite el acceso completo. No puede ser usado para leer y escribir, debido a la necesidad de mantener la coherencia de caché, lo cual sería imposible si el recurso se pusiera a disposición de cualquier forma.

Así mismo DRBD soporta tres modos de replicación, lo cual se detalla en tres modos de sincronización.

- **Protocolo A:** Protocolo de replicación asíncrona. Las operaciones locales de escritura en el nodo principal se consideran terminadas tan pronto como la escritura en el disco local ha finalizado, y el paquete de replicación se ha colocado en el buffer de envío local TCP.
- **Protocolo B:** Protocolo de replicación de memoria síncrona (semi-síncrona). Las operaciones locales de escritura en el nodo principal se consideran terminadas tan pronto como la escritura del disco local se ha producido y el paquete de replicación ha alcanzado el nodo par (peer node).
- **Protocolo C:** Protocolo de replicación síncrona. Las operaciones locales de escritura en el nodo principal se consideran completas solo después de que tanto a nivel local como remoto, la escritura se ha confirmado. Como resultado, al perder un solo nodo se garantiza que no producirá ninguna pérdida de datos.

### 5.2.2. Corosync/OpenAIS

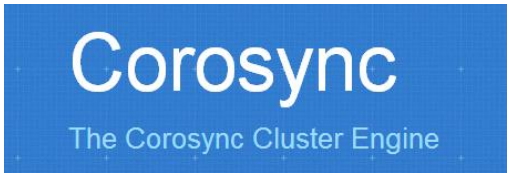



*Figura 18. Logotipo Corosync. Fuente: http://corosync.github.io/corosync/*

Corosync, producto derivado del proyecto OpenAis, es un sistema de comunicación de grupo, con características adicionales para la implementación de alta disponibilidad dentro de las aplicaciones que se dispongan en el clúster. Mantiene las membresías de cluster con las cuales se construyen arreglos de nodo en sincronía.

### 5.2.3. Pacemaker


*Figura 19. Logotipo Pacemaker. Fuente: http://clusterlabs.org/wiki/Pacemaker*

Pacemaker es un software gestor de recursos de clúster (CRM) para plataformas Linux. Logra la máxima disponibilidad de sus servicios de clúster mediante la detección y la recuperación de los nodos y los fallos a nivel de recursos, haciendo uso de la mensajería y capacidades de composición proporcionada por la infraestructura de clúster que se prefiera (ya sea Corosync/OpenAIS o Heartbeat[13]).

[13] http://www.linux-ha.org/wiki/Heartbeat

Pacemaker puede hacer esta labor para clústeres de cualquier tamaño y viene con un modelo de dependencia potente que permite al administrador expresar con precisión las relaciones (tanto orden y ubicación) entre los recursos del clúster.

Algunas de las características más relevantes de Pacemaker son:

- Detección y recuperación de nodo y fallos a nivel de servicio (recursos).
- Almacenamiento agnóstica, sin necesidad de almacenamiento compartido.
- Recurso agnóstico, algo que puede ser escrito puede ser agrupado.
- Soporte de clústeres grandes y pequeños.
- Soporte para cualquier configuración de redundancia entre ellas: Activo / Activo, Activo / Pasivo, N + 1, N + M, N - 1 y N a N.
- Opcionalmente garantiza la integridad de los datos con STONITH (Shoot The Other Node In The Head). Es una técnica que permite asegurar que un servidor que esté en un estado de fallo no interfiera con el funcionamiento del clúster.
- La capacidad de especificar todo el clúster en cuanto a orden, colocación y anti-colocación de los servicios.
- Tipos de recursos avanzados.
- Soporte de servicios que necesitan estar activos en varios nodos.
- Soporte de servicios con múltiples modos (por ejemplo, maestro / esclavo, primario / secundario).
- Unificado, configurable mediante scripts, clúster por comandos Shell.

### 5.3. Selección herramienta de datos compartidos en red

Para la elección de una herramienta que permitiera compartir datos a través de la red, fue necesario indagar e investigar tres tipos de herramientas funcionales para dicho proceso.

A continuación se presenta una comparación de dichas herramientas y el porqué de la elección tomada.

Tabla 1: Comparación herramientas datos compartidos en red.

| Herramienta | Versión | Licencia | Características | | | |
|---|---|---|---|---|---|---|
| | | | Requerimientos Especiales | Nivel de Soporte | Referencias y Popularidad | Compatibilidad con el entorno |
| OCFS | 2.0 | GPL | si | continuo | Muy popular en entornos empresariales | Si |
| NFS | 3.0 | GPL v2 | no | no muy continuo | Incluido en distribuciones LINUX | Si |
| GlusterFS | 3.3 | GPL v3 | no | continuo | Popular en entornos académicos | Si |

Fuente: Los autores

En primera instancia se descartó la herramienta OCFS porque requería hardware especial de sistemas de almacenamiento en red como SAN[14] que funciona con tecnología iSCSI[15]. La infraestructura actual no cuenta con el soporte para este tipo de almacenamiento.

El sistema de archivos NFS está dividido en dos partes principales, el servidor, implementado mediante un paquete extra de software y el cliente que viene incorporado por defecto en la instalación de Debian. El servidor funciona a través

[14] http://es.wikipedia.org/wiki/Red_de_área_de_almacenamiento
[15] http://es.wikipedia.org/wiki/Small_Computer_System_Interface

de un sistema de mapeo de puertos (portmap) que asigna un número de puerto aleatorio para cada servicio, esto imposibilita la aplicación de un filtrado a través de un firewall y deja una importante falla de seguridad. Además de ello en el lado del cliente, NFS presenta un consumo de recursos excesivo y esto lo hace un poco ineficiente y demorado a la hora de transferir datos a través de la red. Por tal razón también se descartó esta opción.

La última herramienta analizada es GlusterFS y es la elegida pues no requiere hardware especial, además que es una herramienta muy usada en entornos académicos y tiene soporte continuo y constante. GlusterFS presenta un sistema de almacenamiento en red basado en NAS[16] que es una tecnología de almacenamiento con capacidad de compartir las unidades, un menor coste, utilización de la misma infraestructura de red y una gestión más sencilla respecto a SAN. Soporta distintos modos de operación que permiten elegir entre distintos métodos de almacenamiento según las características del proyecto en el que se implementa.

Para el desarrollo de este proyecto se eligió el método de volúmenes distribuidos, el cual consiste en distribuir al azar los archivos en bloques a través del servidor. Se hace necesario utilizar este tipo de distribución pues permite que el almacenamiento sea escalable y le da menos importancia a la redundancia de datos tal como se espera en el modelo desarrollado actualmente. La siguiente figura detalla el concepto de volumen distribuido para el compartimiento de archivos a través de la red.

[16] http://es.wikipedia.org/wiki/Network-attached_storage

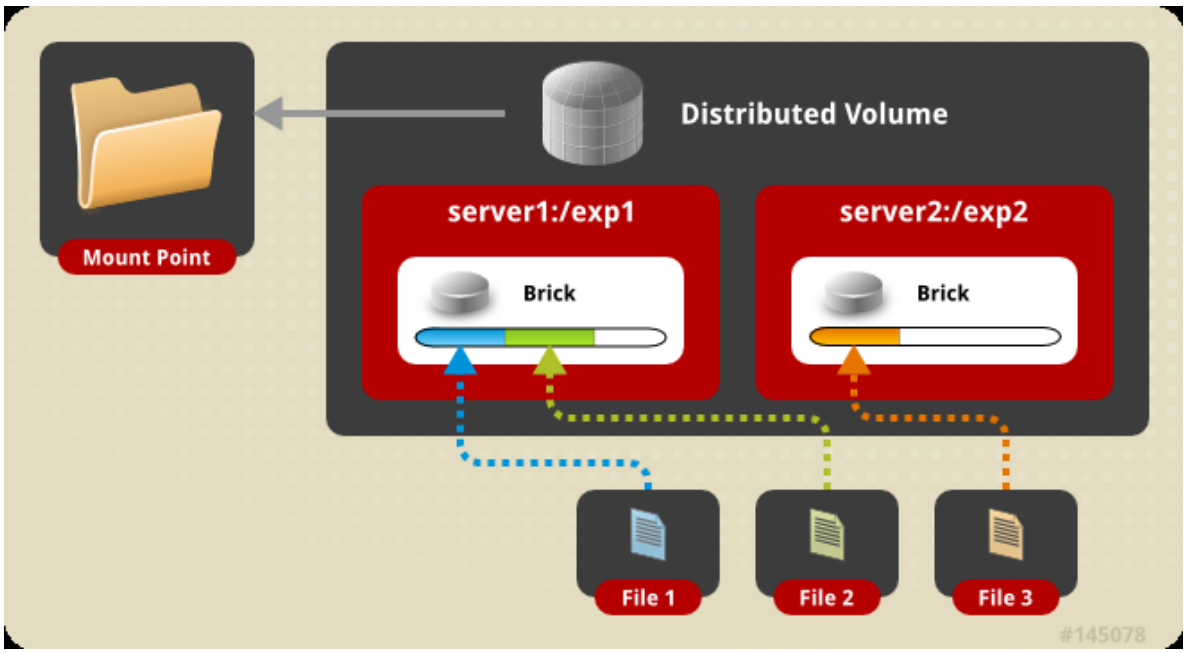



*Figura 20. Volúmenes Distribuidos. Tomado de: Gluster_File_System-3.3.0-Administration_Guide-en-US.pdf p.10*

### 5.4. Selección herramienta de sincronización

Además de cumplir la labor como herramienta de datos compartidos a través de la red por medio de los volúmenes distribuidos, GlusterFS también realiza la función de sincronización de datos a través de los volúmenes replicados, este tipo de volúmenes fue el usado para mantener los datos redundantes en las máquinas virtuales. La función de los volúmenes replicados es crear una copia de los archivos y ubicarlos en múltiples bloques en cada uno de los servidores y de esta manera se garantiza la sincronización de los mismos que se puede observar en la siguiente figura.

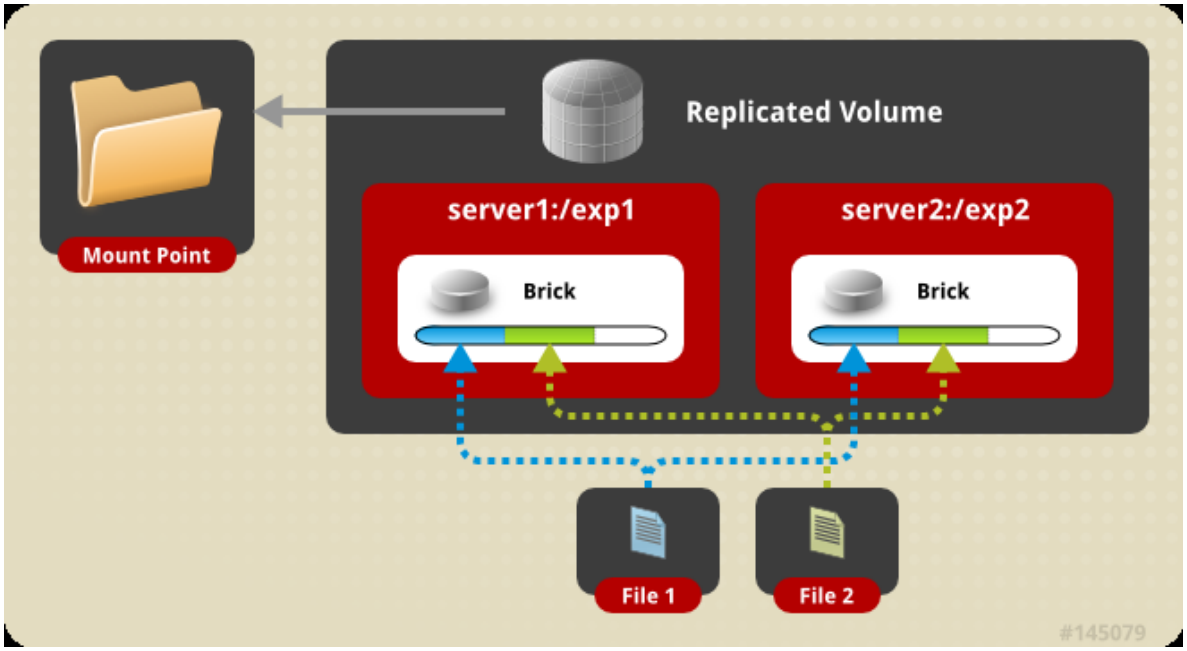



*Figura 21. Volúmenes Replicados. Tomado de: Gluster_File_System-3.3.0-Administration_Guide-en-US.pdf p.12*

### 5.5. Selección herramienta de balanceo de carga

El balanceo de carga se hace necesario implementarlo por medio de alguna herramienta para no saturar un solo servidor con las peticiones hechas por los usuarios. Para la elección de la herramienta se investigaron dos tipos distintos, la primera llamada Nginx[17] un servidor web ligero de alto rendimiento y un proxy para protocolos de correo electrónico. La segunda llamada Zen Load Balancer[18] es un balanceador de carga implementado como un kernel funcional basado en Debian. Tiene una interfaz de administración gráfica y menús con opciones para una implementación rápida.

[17] http://es.wikipedia.org/wiki/Nginx
[18] http://www.zenloadbalancer.org/web/index.php?page=zen-load-balancer-administration-guide

Se elige Zen Load Balancer pues admite balanceo de tráfico de red de cualquier tipo TCP/UDP en cambio Nginx solo admite balanceo de tráfico http, esto lo restringe mucho para el propósito de este proyecto.

Zen Load Balancer posee una interfaz muy amigable con el administrador pues es muy intuitiva y de fácil manejo, mediante las opciones incorporadas se puede programar el balanceo y sus opciones son sencillas y muy gráficas. El balanceo se realiza por medio de un algoritmo llamado Round Robin quien se encarga de dirigir el tráfico equitativamente entre los tres servidores.

En la siguiente figura se puede observar una vista general de la interfaz de administración del Zen Load Balancer.


*Figura 22. Interfaz de administración del ZEN Load Balancer. Fuente: Los Autores*

## 5.6. Implementación modelo de alta disponibilidad

Durante el proceso de implementación del sistema de alta disponibilidad, uno de los mayores obstáculos fue la presencia de servicios corriendo bajo los equipos servidores, pertenecientes a la implementación surgida como resultado del proyecto anterior. Gracias al enfoque basado en componentes asimilado desde la construcción de los modelos, no se hizo necesario aplicar interrupciones sobre los servicios en ejecución. Analizando la implementación previa de este sistema presente en los servidores solo fue necesario aplicar cambios a las configuraciones y actualizaciones de paquetes en base al nuevo enfoque.

Como se quieren mantener las configuraciones y los cambios hechos a nivel de sistema operativo dentro de cada una de las máquinas, se optó por hacer una replicación de los discos virtuales de las máquinas mediante un dispositivo DRBD. De esta forma al momento de ejecutar una instancia en alguno de los hosts, se conservan los cambios hechos previamente por su máquina gemela. Para controlar el encendido y apagado automático de estas máquinas ante cambios indeseados o fallas de los equipos, se realizaron las debidas configuraciones sobre el conjunto de paquetes Pacemaker/Corosync. Estos usan scripts funcionales especiales llamados Resource Agents (RA) que les permite controlar muchos servicios compatibles. El inconveniente presentado fue la ausencia de un RA dentro de la herramienta para controlar máquinas virtuales creadas mediante VirtualBox. La solución surgió desde la misma herramienta al tener la capacidad de soportar RAs personalizados, basados en estándares de escritura de scripts como lo son LSB[19] y OCF[20]. Con esto, el siguiente paso fue crear un script especial usando LSB debido a que es un estándar muy sencillo de entender y es

[19] Siglas de Linux Standard Base, http://es.wikipedia.org/wiki/Linux_Standard_Base
[20] Siglas de Open Cluster Framework, http://en.wikipedia.org/wiki/Open_Cluster_Framework

muy usado en sistemas Linux para crear scripts de inicio. De esta forma se dispuso de un medio para controlar adecuadamente las máquinas virtuales.

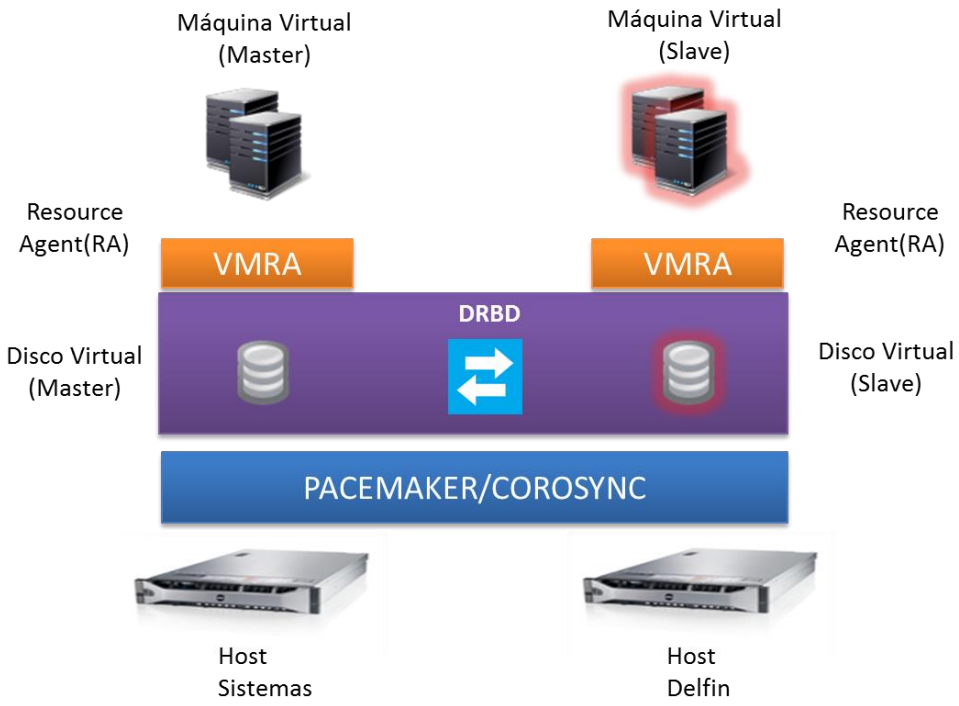



*Figura 23. Modelo de alta disponibilidad actual. Fuente: Los Autores*

El esquema final cuenta con las siguientes partes:

*Hosts*: Equipos servidores físicos. Uno de ellos puede ejecutar las máquinas virtuales si el CRM les asigna el rol de Master(maestro), de lo contrario permanecerá a la espera como Slave(Esclavo) con los servicios apagados.

*Pacemaker/Corosync*: Agente de control de recursos (CRM) usado para la gestión automática de las máquinas virtuales bajo alta disponibilidad. Es el encargado de llevar a cabo acciones como el apagado, encendido y migración de los recursos virtuales a su disposición.

*VMRA*: Virtual Machine Resource Agent por sus siglas en inglés, es el agente de recursos (RA) utilizado por el CRM para administrar las máquinas virtuales.

Máquina virtual: Componente que hospeda y ejecuta los servicios de usuario.

*DRBD*: Dispositivo de replicación que copia a modo de espejo los cambios hechos sobre el disco virtual en uno de los host hacia el host de respaldo. Funciona bajo un esquema Activo/Pasivo.

*Disco Virtual*: Almacena el sistema operativo, los datos y las configuraciones que utilizan las máquinas virtuales para iniciar y funcionar.

## 5.7. Implementación modelo computación en la nube

En la implementación del modelo, se tomó el esquema base que estaba en funcionamiento sobre los equipos servidores. Gracias a la propiedad de modularidad con la que ya contaba, fue posible aplicar cambios sin interrumpir los servicios en funcionamiento. Se aplicó una actualización significativa sobre el Frontend, obteniendo en el proceso mejoras con respecto a la funcionalidad y rendimiento.

En el esquema se dispuso el tercer nodo como almacén de imágenes, esto debido a que no se contaba con un almacén de datos dedicado. Se realizaron configuraciones con paquetes de software especiales para poder compartir las imágenes entre todos los hosts. Este nodo también se incorporó a la infraestructura como host para la ejecución de instancias virtuales.

Con el sistema de alta disponibilidad implementado, se procedió a colocar el Frontend en redundancia asegurándolo de esta forma frente a la caída del host que lo hospeda.

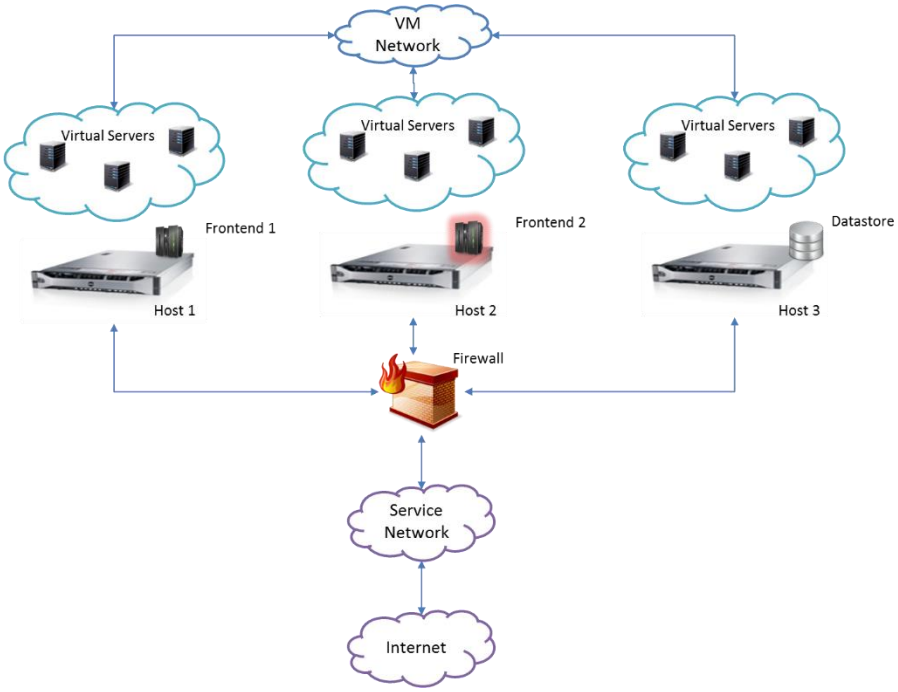



*Figura 24. Modelo de computación en la nube actual. Fuente: Los Autores*

En la figura se muestra el modelo adaptado a la infraestructura.

*Frontend*: Equipo servidor que administra y asigna instancias virtuales a los hosts. También actúa como punto de entrada de paquetes de red a la infraestructura. Se acopla sobre el sistema de alta disponibilidad con el fin de mantenerse en constante funcionamiento, migrando entre los dos hosts según lo requiera.

*Datastore*: Almacena las imágenes virtuales. Fue incorporado dentro del tercer host en un sistema de archivos compartido en red para su acceso de cualquiera de los nodos.

*Virtual Servers*: Son las instancias virtuales en ejecución asignadas previamente por el frontend. Con la incorporación del tercer host, se agregó un nuevo nodo para el hospedaje de más máquinas.

*Service Network*: Permite la conexión de toda la infraestructura con la red universitaria. Todos los equipos servidores cuentan con acceso a ella.

*VM Network*: Red interna entre las instancias virtuales y entre los nodos. Permite el paso de las imágenes almacenadas en el tercer nodo a los demás nodos. Cuenta con una velocidad de 1 Gbps.

## 5.8. Implementación modelo de balanceo y distribución de recursos

Para la implementación de este modelo se partió de la disposición de 3 máquinas virtuales. El objetivo era aprovechar la presencia de los 3 hosts para mantener un servicio con redundancia a tercera escala. De esta forma se distribuiría la carga en cada uno de los equipos.

*Redundancia de datos*

La redundancia, a diferencia del modelo de alta disponibilidad, se da a nivel de los datos y de la base datos, esto con miras a albergar y soportar servicios web. Los datos se sincronizan entre cada una de las máquinas que conforman el clúster y es posible hacer cambios simultáneos sobre el mismo espacio manteniendo la consistencia. Gluster fue la tecnología que permitió dicha sincronización. La base de datos requirió de un tratamiento especial. Debido a la forma en que está construida, no es posible mantenerla en redundancia con acceso simultaneo desde cada de las máquinas. Se aisló la base de datos y se estableció un sistema maestro/esclavo para de esta forma brindar alta disponibilidad y permitir el acceso simultáneo y concurrente.

*Balanceo de carga*

El balanceador de carga fue incorporado como una máquina virtual que redirigía el tráfico proveniente del frontend hacia las máquinas adecuadas.

Este esquema permitió la migración de los servicios que aun funcionaban directamente sobre los equipos servidores. Completando con la modularidad de los sistemas. Los usuarios se conectan a través de una sola dirección de red y desconocen sobre qué máquina están trabajando.

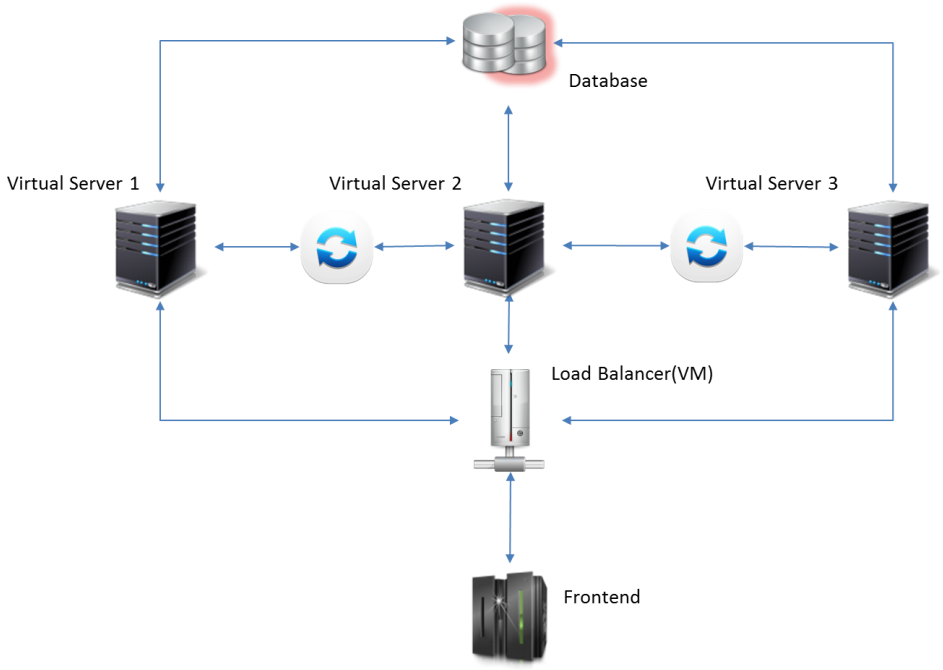



*Figura 25. Modelo de balanceo y distribución de recursos actual. Fuente: Los Autores*

En la figura 25 se observa el esquema del modelo de balanceo de recursos:

*Load Balancer*: Implementado mediante una máquina virtual, redirige el tráfico desde el frontend hacia las máquinas en balanceo. Puede mantener sesiones TCP para evitar que el usuario sea enviado a una máquina distinta durante su trabajo.

*Virtual Server*: Máquinas con sincronización de datos. Reciben y envían las peticiones de usuario que llegan desde el balanceador de carga. Acceden a la base de datos a través de una dirección IP fija.

*Database*: La base datos en alta disponibilidad. Implementada mediante 2 máquinas virtuales en esquema maestro/esclavo. El sistema mantiene una IP de servicio sobre la cual atiende peticiones y que puede rotar entre una máquina u otra según lo requiera.

*Frontend*: Determina cuando una petición se dirige hacia un servicio con balanceo de carga.

## 5.9. Implementación del sistema de soporte

Para la implementación del modelo de soporte se destinó un equipo de altas capacidades de hardware disponible en el sitio de trabajo del grupo CONUSS. Como su objetivo es apoyar las labores de investigación del grupo y funcionar como punto de extensión y reparación de los sistemas en producción, se dotó con una copia de los servicios virtuales con los que cuenta la infraestructura en producción, incluyendo copias de máquinas de administración como el frontend y el balanceador de carga. De esta forma pueden crearse nuevos servicios que son completamente compatibles con el esquema de producción, además de reparar y aplicar mejoras a los existentes.

## 5.10. Migración de servicios

Con el fin de establecer completamente la infraestructura en el enfoque de módulos virtualizados tomado desde la fase de análisis y diseño, se decidió hacer una migración de los servicios que hasta el momento funcionaban directamente sobre el sistema operativo de los equipos servidores que conforman el cluster. Estos estaban conformados principalmente por las aplicaciones MEIWEB y Moodle. Para su traslado a las plataformas virtuales, inicialmente se dedujeron los requisitos de hardware y software necesarios para su funcionamiento. A partir de esta información se crearon las máquinas base con sistema operativo Debian y la cantidad de recursos (RAM, CPU) suficientes para albergarlas. También se dotaron dichos entornos con los paquetes de software requisitos, obtenidos en la fase anterior.

En el sistema base, las aplicaciones funcionaban sobre el modelo de alta disponibilidad que les otorgaba una capa de tolerancia a fallos para así asegurar la continuidad de los procesos que ejecutaban. En el modelo establecido en la fase de diseño, se ubicaron sobre el sistema de balanceo de carga propuesto. Debido a

que son dos de los servicios que más peticiones de usuario recolectan en el periodo de tiempo de un día, el sistema de balanceo puede proporcionar un medio que distribuya la carga ejercida por la demanda de recursos asociada. De esta forma se mantiene la disponibilidad de los servicios y se agrega una característica de distribución de recursos para hacer frente a momentos de alta concurrencia en el sistema.

# 6. PUESTA EN MARCHA DEL MODELO

Después de pasar por las fases de análisis de la infraestructura, del diseño y la implementación del modelo, se presenta en el siguiente apartado la puesta en marcha del modelo.

## 6.1. Servicios virtualizados

Para hacer uso de los servicios virtualizados se determinó que la forma más apropiada de iniciar rápidamente el catálogo de servicios era utilizar imágenes de servicios virtuales que ya existieran en la red, para este fin se utilizó el repositorio de imágenes de TurnKey Linux[21], este es un proyecto de código abierto que ha desarrollado una biblioteca de imágenes de servidores basada en las distribuciones de Linux Debian[22] y Ubuntu[23]. Cada una de las imágenes puede ser instalada en servidores reales y también ofrecen versiones optimizadas para uso como servidores virtuales, al ser soluciones listas para usar.

### 6.1.1. Biblioteca de imágenes de servidores virtuales

En la biblioteca de TurnKey Linux se encuentra un repositorio de 106 imágenes de diferentes servidores virtualizados, las cuales se han descargado y se encuentran disponibles en el repositorio de imágenes establecido en el servidor de almacenamiento para uso del administrador y así poder ofrecer una mejor oferta a los usuarios para sus futuros trabajos. A continuación se presentan algunas de las imágenes más usadas:

[21] http://www.turnkeylinux.org
[22] http://www.debian.org
[23] http://www.ubuntu.com


Joomla[24]: Es un motor de portales dinámicos y gestor de contenidos. Es de código abierto, escrito en PHP.


Moodle[25]: Es una plataforma de aprendizaje en línea (e-learning), también conocido como sistema de administración de cursos de código abierto.


Tomcat[26]: Es un servidor web y un contenedor de servlets[27] y JavaServer Pages[28] de código abierto, esta imagen contiene solo el servidor Tomcat.


Ruby on Rails[29]: Es un framework para aplicaciones web de código abierto, enfocado en el desarrollo ágil de aplicaciones.


LAMP: Es un acrónimo de la integración de tecnologías, estas son: el sistema operativo linux, el servidor web apache, el motor de base de datos MySQL y el lenguaje de programación interpretado PHP.

[24] http://www.joomla.org
[25] http://moodle.org
[26] http://tomcat.apache.org
[27] http://es.wikipedia.org/wiki/Servlet
[28] http://es.wikipedia.org/wiki/JavaServer_Pages
[29] http://rubyonrails.org


MySQL: Es un sistema de administración de bases de datos relacionales de código abierto ampliamente usado en desarrollo de aplicaciones web.


PostgreSQL: Es un sistema de administración de bases de datos objeto-relacionales de código abierto que implementa la mayoría del estándar SQL: 2008.


LAPP: Es un acrónimo de la integración de tecnologías, esta son: el sistema operativo Linux, el servidor web apache, el motor de base de datos PostgreSQL[30] y el lenguaje de programación interpretado PHP.


Apache Tomcat: Es un servidor web y un contenedor de servlets y JavaServer Pages de código abierto, se diferencia a la versión Tomcat en que integra como servidor web a Apache[31].

Además existe un servicio más personalizado creado para los usuarios donde cada uno puede elegir a su gusto y necesidad las características de los servicios que necesite y así desarrollar sus propios proyectos empezando con plantillas base y no pre configuradas como las mencionadas anteriormente.

[30] http://www.postgresql.org
[31] http://httpd.apache.org

## 6.2. Copias de seguridad

Las copias de seguridad en todos los aspectos críticos para la infraestructura y el modelo son fundamentales pues se logra tener un respaldo de la información en caso que ocurra algo inesperado como daño en la misma o inconsistencias en el sistema. La forma de sacar los backups o copias de seguridad cambió respecto a la forma como lo realizaban anteriormente. Ahora se realizan tres tipos de copias de seguridad por separado para una mejor administración y facilidad de manejo, las cuales se enuncian a continuación.

### 6.2.1. Copia de seguridad del sistema

El sistema operativo es una capa multifuncional que se acopla con una arquitectura de hardware con el fin de gestionar los recursos que tiene a su disposición. Sirve como base para la implementación de servicios mediante paquetes de software que demandan cierta cantidad de recursos para llevar a cabo la tarea que les ha sido incorporada. Una falla a nivel de sistema operativo podría representar un catastrófico desenlace para las aplicaciones en funcionamiento siendo comprometidos en el proceso datos de suma importancia para los usuarios del sistema. Es por eso que se plantea realizar periódicamente copias de seguridad que brinden una herramienta eficaz para reducir el tiempo de recuperación de la infraestructura. Mediante el enfoque abordado en este proyecto, se disminuye la cantidad de servicios dependientes de forma directa del sistema operativo, por tanto se facilita la recuperación en un escenario de fallas. Las copias seguridad se realizan periódicamente y cumplen la función de asegurar las configuraciones principales establecidas durante el proceso de implementación.

### 6.2.2. Copia de seguridad de máquinas virtuales

Como se enunciaba anteriormente gracias al enfoque de componentes y a la propiedad modular de los servicios, se pueden realizar copias de seguridad de las máquinas virtuales con un grado de sensibilidad alto. Gracias a las opciones dispuestas por el hipervisor, este proceso se realiza de forma rápida y confiable. Las copias se establecen a través de un .comprimido que se exporta de la máquina y es dispuesto en áreas especiales con suficiente espacio para ello.

A diferencia de los respaldos a nivel de sistema operativo, las copias de seguridad de servicios virtuales pueden recuperar componentes individuales dañados sin afectar el sistema general o los demás componentes hospedados en el mismo entorno.

### 6.2.3. Copia de seguridad de los archivos de configuración

Se cuenta también con un repositorio de todos los archivos de configuración de servicios y hosts. De una forma más específica a la planteada en los tipos de copias de seguridad anteriores, los respaldos a nivel de archivos de configuración pretenden brindar una herramienta que soporte el cambio continuo en el sistema ya sea por mejoras o la incorporación de nuevos servicios. Este es un proceso que manualmente se realiza extrayendo los archivos modificados desde los servidores en producción. De esta forma se tiene base documental que permita establecer puntos de fallo debido a modificaciones. También se evita recurrir a recuperaciones de mayor envergadura ya sea a nivel de sistema o de servicios virtuales.

### 6.2.4. Copia de seguridad de los datos

Se continúa con el esquema que venía en funcionamiento desde el modelo anterior manteniendo un backup completo semanal y los días restantes se crean backups diferenciales uno cada día. La figura 26 describe el esquema de copias de respaldo implementado en los equipos.

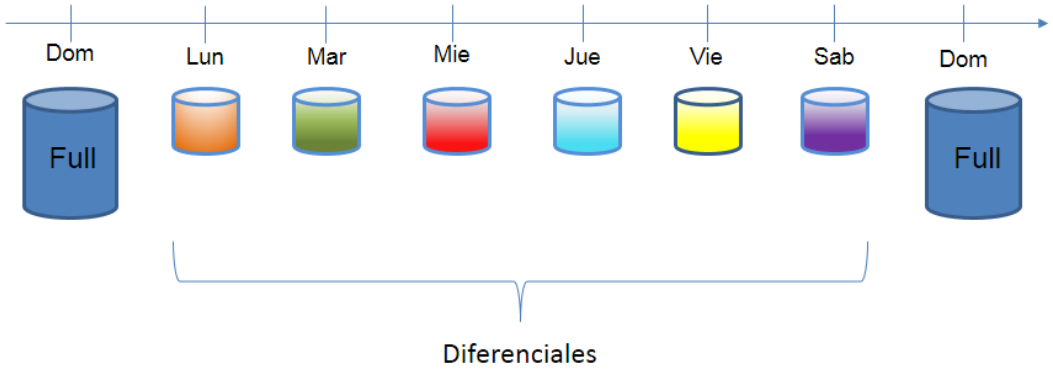



*Figura 26. Esquema de respaldos periódicos locales. Fuente: Los Autores*

Al implementarse este esquema y luego de una serie de pruebas de funcionamiento realizadas en un entorno de máquinas virtuales, se infiere que el esquema funciona adecuadamente en el nuevo entorno de trabajo. Se diseñó un script complementario que automatiza la rotación de archivos en base a la política de hospedaje de backups acordada por el grupo. De esta forma se optimiza el espacio de las áreas de almacenamiento dedicadas.

## 6.3. Seguridad

Esta labor es entendida como un proceso continuo que involucra aspectos como el monitoreo administrativo continuo del sistema para visualizar los puntos más vulnerables y el fortalecimiento de estos, con el fin de reducir las oportunidades de ataque y por consiguiente lograr que la probabilidad de vulnerar el sistema sea menor.

### 6.3.1. Actualizaciones de seguridad

Se decidió aplicar cierta automatización a este proceso que venía haciéndose de forma manual. Se creó un servicio virtual especial que actúa como repositorio de actualizaciones de los sistemas y componentes basados en Debían. Dicha máquina virtual también desempeña otras labores de seguridad que serán mencionadas más adelante. Establece una sincronización con los repositorios oficiales de seguridad esta distribución. Periódicamente comprueba el estado de actualizaciones de los paquetes y descarga según lo requiera los archivos necesarios para aplicar parches. Las demás instancias virtuales apuntan al repositorio local, evitando realizar una conexión a los repositorios externos.

Para hacer uso de estas actualizaciones solo basta con transmitirlas desde el monitor a la máquina virtual respectiva, esta tarea se hace de forma automática y sincronizada. La funcionalidad de los paquetes de software no se ve afectada debido a que no realizan cambios de versión en el proceso. Con esto se quiere evitar el uso excesivo de la red descargando constantemente las actualizaciones, al tener un repositorio local de las mismas, optimizando y reduciendo tiempos en el proceso de actualización, además de ser mucho más rápida la transferencia por medio local que por la red.

### 6.3.2. Sistemas de control y monitoreo

Como la totalidad de los accesos a cada servidor se hace a través de la red es muy importante que cada equipo cuente con una protección de cortafuegos para limitar el acceso por puertos no deseados, así como accesos por protocolos de comunicación autorizados, entre otras características definibles según sea la herramienta a utilizar como firewall.

El sistema operativo GNU/Linux cuenta con su sistema cortafuegos instalado por defecto llamado IPtables, el cual se encuentra configurado respectivamente en cada equipo y permite gestionar el tráfico de red de entrada y salida del sistema. EL frontend utiliza este sistema con el fin de centralizar el control de acceso de los paquetes desde y hacia las instancias virtuales. De esta forma se simplifica la seguridad de cada máquina y facilita la labor administrativa.

Por otra parte, gracias a los paquetes de monitoreo se detectan varios intentos de intrusión por parte de equipos externos, los cuales podrían causar daño en el modelo y en los sistemas. Es por ello que se usan herramientas que además de reportar los ataques realizan una acción preventiva que bloquean dichos accesos las cuales son fail2ban[32] y logwatch[33]; fail2ban es una herramienta para la prevención de intrusiones, que permite por medio reglas preestablecidas sobre los registros del sistema, realizar tareas de bloqueo y generación de alertas. Cubre áreas como el fallo de autenticación de usuarios hasta análisis de peticiones al servidor web apache. Para el uso de la herramienta se optimizan algunos de los filtros que incluye por defecto y se generan otros adecuándolos a los objetivos del proyecto. Logwatch es una herramienta de análisis de los archivos de registro del

[32] www.fail2ban.org
[33] www.logwatch.org

sistema, permite enviar un resumen diario de las diferentes incidencias que se presentan sobre el servidor y en casos puntuales produce advertencias extras sobre dichos resultados.

También se dispone de una herramienta para vigilar el monitoreo de la red llamada Ganglia[34], la cual es un sistema de monitoreo distribuido escalable para sistemas de cómputo de alto desempeño que permite llevar registros históricos de diferentes parámetros del sistema como el monitoreo recursos de CPU, RAM, red etc, y los representa gráficamente. Este sistema fue incorporado dentro una máquina virtual que hace las veces de monitor, obteniendo los datos estadísticos de las demás máquinas.

Finalmente y no menos importante se hace uso del manual de seguridad de debian del cual se extraen pautas de seguridad para aplicarlos en el sistema. También se hace uso de algunos apartes de la norma técnica colombiana NTC/ISO 27002 [6] que describe un código de práctica para la gestión de la seguridad.

### 6.4. Documentación administrativa

Con el fin de mantener la gestión administrativa de los nuevos servicios, se actualizaron y modificaron los manuales de administración adecuando al nuevo modelo y al principio seguido durante la implementación. Se creó un esquema documental que sigue el principio de diseño establecido para la infraestructura. Ya no existe un manual general que consigna todas las características de los sistemas. En el nuevo esquema se escriben documentos individuales, uno por cada servicio con independencia suficiente para abstraer por separado sus

[34] http://ganglia.sourceforge.net/

configuraciones. Procesos y configuraciones generales y aplicables a todos los servicios se detallan por aparte. De esta forma surge una jerarquía que parte de la documentación de los servidores y los sistemas base y termina en documentos más pequeños que detallan la implementación y funcionamiento de servicios individuales.


*Figura 27. Esquema jerárquico de documentación. Fuente: Los Autores*

## 6.5. Pruebas de funcionamiento

En este apartado se presentan las pruebas que miden los tiempos de inicio de las máquinas instanciadas, el tiempo que requieren para migrar a otro host, el tiempo de migración del frontend cuando se cae un nodo, además de dos pruebas que se realiza con el balanceador de carga.

### 6.5.1. Pruebas de instanciación de imágenes virtuales

Para la realización de las pruebas de instanciación se tomó el registro producido por la herramienta durante el proceso de lanzamiento de cada máquina, este

registra inicialmente con una marca de tiempo el momento en que la instancia pasa por cuatro estados, ACTIVE, PROLOG, BOOT y RUNNING. Los estados que interesan para este análisis es el tiempo que toma para el sistema pasar del estado ACTIVE, momento en el cual se inicia la instanciación, al estado BOOT, momento de espera para que el hipervisor cree la máquina y de este al estado RUNNING, punto en el cual la máquina ya se encuentra en un estado utilizable por el usuario final, con estos valores se determina el tiempo promedio de lanzamiento de cada máquina en los tres hosts. Para este proceso se instanciaron 30 máquinas con imágenes LAMP[35], los datos obtenidos se presentan en la tabla 2.

[35] http://es.wikipedia.org/wiki/LAMP

*Tabla 2. Tiempo de duración del proceso de instanciación.*

| Host | MV_ID | Hora | Estado | Hora | Estado | Duración de Copiado | Hora | Estado | Duración Total |
|---|---|---|---|---|---|---|---|---|---|
| Delfín | 45 | 09:10:46 | ACTIVE | 09:10:56 | BOOT | 0:00:10 | 09:11:09 | RUNNING | 0:00:23 |
| Sistemas | 46 | 09:11:16 | ACTIVE | 09:11:25 | BOOT | 0:00:09 | 09:11:41 | RUNNING | 0:00:25 |
| Delfín | 47 | 09:12:16 | ACTIVE | 09:12:25 | BOOT | 0:00:09 | 09:12:39 | RUNNING | 0:00:23 |
| Sealion | 48 | 09:13:46 | ACTIVE | 09:13:47 | BOOT | 0:00:01 | 09:14:00 | RUNNING | 0:00:14 |
| Sistemas | 49 | 09:14:46 | ACTIVE | 09:14:55 | BOOT | 0:00:09 | 09:15:10 | RUNNING | 0:00:25 |
| Delfín | 50 | 09:15:16 | ACTIVE | 09:15:25 | BOOT | 0:00:09 | 09:15:39 | RUNNING | 0:00:23 |
| Sealion | 51 | 09:16:16 | ACTIVE | 09:16:17 | BOOT | 0:00:01 | 09:16:29 | RUNNING | 0:00:13 |
| Sistemas | 52 | 09:17:16 | ACTIVE | 09:17:25 | BOOT | 0:00:09 | 09:17:41 | RUNNING | 0:00:25 |
| Delfín | 53 | 09:18:46 | ACTIVE | 09:18:56 | BOOT | 0:00:10 | 09:19:12 | RUNNING | 0:00:26 |
| Sealion | 54 | 09:19:46 | ACTIVE | 09:19:47 | BOOT | 0:00:01 | 09:19:59 | RUNNING | 0:00:13 |
| Sistemas | 55 | 09:20:46 | ACTIVE | 09:20:55 | BOOT | 0:00:09 | 09:21:12 | RUNNING | 0:00:26 |
| Delfín | 56 | 09:22:16 | ACTIVE | 09:22:25 | BOOT | 0:00:09 | 09:22:39 | RUNNING | 0:00:23 |
| Sealion | 57 | 09:23:46 | ACTIVE | 09:23:47 | BOOT | 0:00:01 | 09:23:59 | RUNNING | 0:00:13 |
| Sistemas | 58 | 09:25:16 | ACTIVE | 09:25:25 | BOOT | 0:00:09 | 09:25:41 | RUNNING | 0:00:25 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Delfín | 59 | 09:26:46 | ACTIVE | 09:26:55 | BOOT | 0:00:09 | 09:27:09 | RUNNING | 0:00:23 |
| Sealion | 60 | 09:59:16 | ACTIVE | 09:59:17 | BOOT | 0:00:01 | 09:59:29 | RUNNING | 0:00:13 |
| Sistemas | 61 | 09:59:46 | ACTIVE | 09:59:55 | BOOT | 0:00:09 | 10:00:11 | RUNNING | 0:00:25 |
| Delfín | 62 | 10:01:46 | ACTIVE | 10:01:55 | BOOT | 0:00:09 | 10:02:09 | RUNNING | 0:00:23 |
| Sealion | 63 | 10:03:16 | ACTIVE | 10:03:17 | BOOT | 0:00:01 | 10:03:29 | RUNNING | 0:00:13 |
| Sistemas | 64 | 10:04:16 | ACTIVE | 10:04:25 | BOOT | 0:00:09 | 10:04:40 | RUNNING | 0:00:24 |
| Delfín | 65 | 10:05:16 | ACTIVE | 10:05:25 | BOOT | 0:00:09 | 10:05:39 | RUNNING | 0:00:23 |
| Sealion | 66 | 10:07:16 | ACTIVE | 10:07:17 | BOOT | 0:00:01 | 10:07:30 | RUNNING | 0:00:14 |
| Sistemas | 67 | 10:08:46 | ACTIVE | 10:08:55 | BOOT | 0:00:09 | 10:09:11 | RUNNING | 0:00:25 |
| Delfín | 68 | 10:12:46 | ACTIVE | 10:12:56 | BOOT | 0:00:10 | 10:13:09 | RUNNING | 0:00:23 |
| Sealion | 69 | 10:15:16 | ACTIVE | 10:15:17 | BOOT | 0:00:01 | 10:15:29 | RUNNING | 0:00:13 |
| Sistemas | 70 | 10:18:16 | ACTIVE | 10:18:25 | BOOT | 0:00:09 | 10:18:42 | RUNNING | 0:00:26 |
| Delfín | 71 | 10:20:16 | ACTIVE | 10:20:25 | BOOT | 0:00:09 | 10:20:39 | RUNNING | 0:00:23 |
| Sealion | 72 | 10:21:46 | ACTIVE | 10:21:47 | BOOT | 0:00:01 | 10:21:59 | RUNNING | 0:00:13 |
| Sistemas | 73 | 10:22:46 | ACTIVE | 10:22:55 | BOOT | 0:00:09 | 10:23:11 | RUNNING | 0:00:25 |
| Delfín | 74 | 10:23:46 | ACTIVE | 10:23:55 | BOOT | 0:00:09 | 10:24:10 | RUNNING | 0:00:24 |

Promedio Sistemas y Delfín: 00:00:24 Desv. Estándar Sistemas y Delfín: 1,16700675

Promedio Sealion: 00:00:13 Desv. Estándar Sealion: 0,44095855

Fuente: Los Autores

Los resultados obtenidos muestran dos tiempos promedio, el primero de los servidores Sistemas y Delfín y el segundo del servidor Sealion. Como se observa los tiempos de estos varían en 11 segundos, esto se debe a que Sealion es el nodo de almacenamiento, es decir donde se encuentran las imágenes de las máquinas virtuales por lo tanto la instanciación en los nodos Sistemas y Delfín incurre en un tiempo mayor pues se tiene que hacer paso a través de la red para traer las imágenes, mientras que Sealion lo hace directamente y el tiempo total de instanciación se hace mucho menor. Se utilizaron como objeto prueba, máquinas virtuales basadas en una imagen tipo LAMP de 934 MB y una plantilla que asignaba 256 MB de RAM y 1 núcleo de CPU.

Para Sistemas y Delfín se puede observar que en promedio las 21 máquinas virtuales creadas en estos hosts están listas para ser usadas 24 segundos después de ser instanciadas, mientras que para Sealion las restantes 9 máquinas solo tardan 13 segundos en estar listas para poder hacer uso de ellas.

Por otra parte, analizando los estados por los que pasa la máquina a través de su instanciación se pueden sacar otras conclusiones. Una de ellas surge del primer estado por el que atraviesa, ACTIVE, momento en el que empieza a realizar el copiado de las imágenes del tercer nodo (Sealion). Como se observa en los nodos Sistemas y Delfín, esto conlleva un tiempo no mayor a 10 segundos, lo cual es esperado debido al enlace Gigabit Ethernet utilizado para unir estos nodos, hasta el momento de espera de creación de la máquina por parte del hipervisor indicado en el estado BOOT. Es un tiempo relativamente bajo ya que las imágenes LAMP cuentan con un promedio de 934 Mb de tamaño y el resultado es muy satisfactorio pues su copiado se hace rápida y efectivamente. El tiempo de transición de estos mismos estados en Sealion es solo 1 segundo ya que el proceso de copiado se hace localmente.

Posteriormente viene la transición del estado de creación de la máquina (BOOT) al estado en el que la máquina está lista para ser usada por el usuario (RUNNING), esto también se hace en un tiempo corto en promedio 14 segundos donde prepara lo necesario para que finalmente la máquina quede establecida.
De esta prueba también se puede concluir que la herramienta usada para el compartimiento de datos a través de la red, GlusterFS, hace su trabajo eficientemente y mejora los tiempos que se establecían en la anterior plataforma.

### 6.5.2. Pruebas de migración de instancias virtuales entre los hosts

La migración de instancias virtuales permite trasladar la prestación del servicio que estas almacenan a un nuevo host, ya sea por motivos como la liberación de recursos de hardware así como el respaldo frente a eventos de mantenimiento del host que inicialmente las hospeda.
Para las pruebas de migración los registros de las máquinas instanciadas se agregan dos nuevos estados, SAVE_MIGRATE, en el cual se guarda el estado de la máquina y se apaga y PROLOG_MIGRATE, en el cual los archivos de la instancia virtual son transportados desde el host origen de la migración al host destino. Los estados que son representativos en la prueba de migración son el estado SAVE_MIGRATE y su paso al estado RUNNING.

La forma de análisis del proceso de migración que debe ejecutarse sobre una instancia que se encuentre en funcionamiento, se enfocó en el tiempo de respuesta de la instancia, es decir, cuanto tiempo no estará disponible la instancia mientras se realiza la migración.

Para dicha prueba se utiliza una herramienta llamada netcat[36], con la cual se realizan conexiones a diferentes puertos de los protocolos tcp y udp. Se realizan

[36] http://netcat.sourceforge.net

pruebas cada segundo para determinar si la instancia virtual sigue enviando respuesta desde una conexión remota. La tabla 3 muestra los resultados obtenidos.

*Tabla 3. Tiempos de recuperación de la disponibilidad del servicio*

| Fecha | Hora | Resultado netcat | Estado | Tiempo |
|---|---|---|---|---|
| 2013-Abr-26 | 10:51:26 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 10:51:27 | New VM state is | SAVE_MIGRATE | 0:00:01 |
| 2013-Abr-26 | 10:51:28 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 10:52:55 | New VW state is | RUNNING | 0:01:29 |
| 2013-Abr-26 | 10:53:02 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:36** |
| 2013-Abr-26 | 11:07:33 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 11:07:34 | New VM state is | SAVE_MIGRATE | 0:00:01 |
| 2013-Abr-26 | 11:07:35 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 11:08:57 | New VW state is | RUNNING | 0:01:24 |
| 2013-Abr-26 | 11:09:04 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:31** |
| 2013-Abr-26 | 11:37:03 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 11:37:04 | New VM state is | SAVE_MIGRATE | 0:00:01 |
| 2013-Abr-26 | 11:37:05 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 11:38:33 | New VW state is | RUNNING | 0:01:30 |
| 2013-Abr-26 | 11:38:40 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:37** |
| 2013-Abr-26 | 11:42:16 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 11:42:17 | New VM state is | SAVE_MIGRATE | 0:00:01 |
| 2013-Abr-26 | 11:42:18 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 11:43:38 | New VW state is | RUNNING | 0:01:22 |
| 2013-Abr-26 | 11:43:44 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:28** |
| 2013-Abr-26 | 11:47:27 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 11:47:28 | New VM state is | SAVE_MIGRATE | 0:00:01 |
| 2013-Abr-26 | 11:47:29 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 11:48:57 | New VW state is | RUNNING | 0:01:30 |
| 2013-Abr-26 | 11:49:04 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:37** |
| 2013-Abr-26 | 14:24:18 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 14:24:18 | New VM state is | SAVE_MIGRATE | 0:00:00 |
| 2013-Abr-26 | 14:24:20 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 14:25:47 | New VW state is | RUNNING | 0:01:29 |
| 2013-Abr-26 | 14:25:56 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:38** |
| 2013-Abr-26 | 14:28:02 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 14:28:02 | New VM state is | SAVE_MIGRATE | 0:00:00 |
| 2013-Abr-26 | 14:28:04 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 14:29:38 | New VW state is | RUNNING | 0:01:36 |
| 2013-Abr-26 | 14:29:47 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:45** |
| 2013-Abr-26 | 14:32:40 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 14:32:40 | New VM state is | SAVE_MIGRATE | 0:00:00 |

| | | | | |
|---|---|---|---|---|
| 2013-Abr-26 | 14:32:42 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 14:34:11 | New VW state is | RUNNING | 0:01:31 |
| 2013-Abr-26 | 14:34:17 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:37** |
| 2013-Abr-26 | 14:36:45 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 14:36:46 | New VM state is | SAVE_MIGRATE | 0:00:01 |
| 2013-Abr-26 | 14:36:47 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 14:38:24 | New VW state is | RUNNING | 0:01:39 |
| 2013-Abr-26 | 14:38:32 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:47** |
| 2013-Abr-26 | 14:40:01 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 14:40:01 | New VM state is | SAVE_MIGRATE | 0:00:00 |
| 2013-Abr-26 | 14:40:03 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 14:41:32 | New VW state is | RUNNING | 0:01:31 |
| 2013-Abr-26 | 14:41:39 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:38** |
| 2013-Abr-26 | 14:47:30 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 14:47:30 | New VM state is | SAVE_MIGRATE | 0:00:00 |
| 2013-Abr-26 | 14:47:32 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 14:48:58 | New VW state is | RUNNING | 0:01:28 |
| 2013-Abr-26 | 14:49:04 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:34** |
| 2013-Abr-26 | 14:50:21 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 14:50:22 | New VM state is | SAVE_MIGRATE | 0:00:01 |
| 2013-Abr-26 | 14:50:23 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 14:51:56 | New VW state is | RUNNING | 0:01:35 |
| 2013-Abr-26 | 14:52:04 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:43** |
| 2013-Abr-26 | 14:54:25 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 14:54:26 | New VM state is | SAVE_MIGRATE | 0:00:01 |
| 2013-Abr-26 | 14:54:27 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 14:55:55 | New VW state is | RUNNING | 0:01:30 |
| 2013-Abr-26 | 14:56:03 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:38** |
| 2013-Abr-26 | 15:01:01 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 15:01:02 | New VM state is | SAVE_MIGRATE | 0:00:01 |
| 2013-Abr-26 | 15:01:03 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 15:02:29 | New VW state is | RUNNING | 0:01:28 |
| 2013-Abr-26 | 15:02:36 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:35** |
| 2013-Abr-26 | 15:03:43 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | 0:00:00 |
| 2013-Abr-26 | 15:03:44 | New VM state is | SAVE_MIGRATE | 0:00:01 |
| 2013-Abr-26 | 15:03:45 | cloudeisi.uis.edu.co [10.10.20.53] 80 | Connection timed out | 0:00:02 |
| 2013-Abr-26 | 15:05:18 | New VW state is | RUNNING | 0:01:35 |
| 2013-Abr-26 | 15:05:26 | cloudeisi.uis.edu.co [10.10.20.53] 80 | open | **0:01:43** |

Promedio timed out: **0:01:38**

Desviación Estándar: 5,074

Fuente: Los autores

En la prueba, el promedio de tiempo en el cual la instancia corta su comunicación es 98 segundos. A diferencia del proceso de instanciación, el software de administración debe llevar a cabo acciones previas antes del copiado de la imagen entre los nodos. El apagado de la máquina en funcionamiento, el respaldo de la imagen y el borrado de la información de la máquina dentro del hipervisor, hacen parte del proceso de preparación para el traspaso hacia el nuevo nodo. Debido a ello se esperan tiempos más largos en comparación con un proceso de instanciación.

### 6.5.3. Pruebas de migración de máquinas virtuales en alta disponibilidad

La prueba se basa en medir el tiempo en el que la máquina pierde comunicación durante el proceso de migrado. Se utilizó la máquina que hospeda los servicios del frontend debido a su importancia dentro del sistema planteado. El proceso pasa por varias etapas, iniciando con el apagado de la máquina, el desmontaje del sistema de archivos, la promoción del nodo esclavo a nodo maestro, el montaje del sistema de archivos en el nuevo nodo y el posterior encendido de la máquina. En la prueba se toman en cuenta dos tiempos concernientes a dos escenarios. Inicialmente se observa y mide el tiempo en el que el frontend pierde comunicación desde la red exterior, detallado en la Tabla 4. Dichas estadísticas se basan en la información obtenida por la herramienta ping. Como el frontend actúa como intermediario entre la red externa y la red interna que sirve de soporte a las instancias virtuales, se decidió en un segundo escenario medir el tiempo que tarda en restablecerse la comunicación con alguna de las máquinas dentro del cloud, véase Tabla 5.

*Tabla 4. Tiempos de recuperación de la comunicación a través de ping.*

| Hora | Resultado ping | Tiempo |
|---|---|---|
| 00:17:50 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:18:12 | 64 bytes from 10.10.0.1: | 0:00:22 |
| 00:21:02 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:21:21 | 64 bytes from 10.10.0.1: | 0:00:19 |
| 00:24:44 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:25:04 | 64 bytes from 10.10.0.1: | 0:00:20 |
| 00:27:13 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:27:32 | 64 bytes from 10.10.0.1: | 0:00:19 |
| 00:29:46 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:30:06 | 64 bytes from 10.10.0.1: | 0:00:20 |
| 00:32:13 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:32:33 | 64 bytes from 10.10.0.1: | 0:00:20 |
| 00:35:01 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:35:20 | 64 bytes from 10.10.0.1: | 0:00:19 |
| 00:36:55 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:37:15 | 64 bytes from 10.10.0.1: | 0:00:20 |
| 00:38:50 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:39:10 | 64 bytes from 10.10.0.1: | 0:00:20 |
| 00:41:02 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:41:23 | 64 bytes from 10.10.0.1: | 0:00:21 |
| 00:43:44 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:44:05 | 64 bytes from 10.10.0.1: | 0:00:21 |
| 00:46:05 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:46:24 | 64 bytes from 10.10.0.1: | 0:00:19 |
| 00:48:30 | 64 bytes from 10.10.0.1: Timed out | 0:00:00 |
| 00:48:48 | 64 bytes from 10.10.0.1: | 0:00:18 |
| 00:50:55 | 64 bytes from 10.10.0.1: | 0:00:00 |

| | | |
|---|---|---|
| 00:51:14 | Timed out<br>64 bytes from 10.10.0.1: | 0:00:19 |
| 00:54:57 | 64 bytes from 10.10.0.1: | 0:00:00 |
| 00:55:17 | Timed out<br>64 bytes from 10.10.0.1: | 0:00:20 |

Promedio timed out: 0:00:20

Desviación Estándar: 1,014

Fuente: Los Autores

*Tabla 5. Tiempos de recuperación del frontend.*

| Hora | Estado | Tiempo |
|---|---|---|
| 00:17:50 | open | 0:00:00 |
| 00:17:57 - 00:18:11 | Connection timed out | 0:00:21 |
| 00:18:12 | open | 0:00:22 |
| 00:21:01 | open | 0:00:00 |
| 00:21:03 - 00:21:21 | Connection timed out | 0:00:20 |
| 00:21:22 | open | 0:00:21 |
| 00:24:44 | open | 0:00:00 |
| 00:24:46 - 00:25:04 | Connection timed out | 0:00:20 |
| 00:25:05 | Open | 0:00:21 |
| 00:27:13 | open | 0:00:00 |
| 00:27:15 - 00:27:31 | Connection timed out | 0:00:18 |
| 00:27:32 | open | 0:00:19 |
| 00:29:49 | open | 0:00:00 |
| 00:29:51 - 00:30:07 | Connection timed out | 0:00:18 |
| 00:30:08 | open | 0:00:19 |
| 00:32:13 | open | 0:00:00 |
| 00:32:15 - 00:32:33 | Connection timed out | 0:00:20 |
| 00:32:35 | open | 0:00:22 |
| 00:35:02 | open | 0:00:00 |
| 00:35:04 - 00:35:20 | Connection timed out | 0:00:18 |
| 00:35:21 | open | 0:00:19 |
| 00:36:55 | open | 0:00:00 |
| 00:36:58 - 00:37:17 | Connection timed out | 0:00:22 |
| 00:37:18 | open | 0:00:23 |
| 00:38:49 | open | 0:00:00 |

| | | |
|---|---|---|
| 00:38:52 - 00:39:09<br>00:39:10 | Connection timed out<br>open | 0:00:20<br>0:00:21 |
| 00:41:02<br>00:41:04 - 00:41:22<br>00:41:23 | open<br>Connection timed out<br>open | 0:00:00<br>0:00:20<br>0:00:21 |
| 00:43:45<br>00:43:47 - 00:44:05<br>00:44:06 | open<br>Connection timed out<br>open | 0:00:00<br>0:00:20<br>0:00:21 |
| 00:46:04<br>00:46:06 - 00:46:25<br>00:46:26 | open<br>Connection timed out<br>open | 0:00:00<br>0:00:21<br>0:00:22 |
| 00:48:30<br>00:48:32 - 00:48:48<br>00:48:49 | open<br>Connection timed out<br>open | 0:00:00<br>0:00:18<br>0:00:19 |
| 00:50:55<br>00:50:57 - 00:51:13<br>00:51:14 | open<br>Connection timed out<br>open | 0:00:00<br>0:00:18<br>0:00:19 |
| 00:54:56<br>00:54:58 - 00:55:17<br>00:55:18 | open<br>Connection timed out<br>open | 0:00:00<br>0:00:21<br>0:00:22 |
| | Promedio timed out: | 0:00:21 |
| | Desviación Estándar: | 1,387 |

Fuente: Los Autores

En promedio 20 segundos se necesitan para recuperar el funcionamiento del frontend después de un proceso de migrado, tiempo que se considera aceptable para hacer frente a caídas inesperadas del servicio. A diferencia de una migración de instancias, no se realiza una copia del disco ya que este cuenta con redundancia inicialmente. El tiempo se invierte en hacer el cambio y en encender nuevamente la máquina. El tiempo de recuperación de la comunicación con instancias virtuales es prácticamente inmediato en relación con el restablecimiento del frontend.

### 6.5.4. Pruebas del sistema de balanceo de carga

Tal como se detalló en una sección anterior, el sistema de balanceo de carga apoya la ejecución de servicios como lo son las aulas virtuales. Por tal motivo, para poner a prueba el sistema y de paso ver el funcionamiento de dichas aplicaciones en un escenario de producción, se llevaron a cabo actividades experimentales en un entorno académico con estudiantes que usan frecuentemente dichas aplicaciones. Mediante un consenso con el grupo de soporte de las aulas virtuales, se establecieron las pruebas que permitieran evaluar el sistema de balanceo y el funcionamiento de las aplicaciones en el nuevo entorno.

Las mediciones se fundamentaron en la cantidad de peticiones exitosas que llegaban a los servidores web y la distribución de usuarios entre cada una de las máquinas, observando en el proceso el consumo de CPU y RAM asociados.

Real servers status 3 servers, 3 current

| Server | Address | Port | Status | Pending Conns | Established Conns | Closed Conns | Clients |
|---|---|---|---|---|---|---|---|
| 0 | 10.10.8.1 | 443 | | 0 | 0 | 0 | 4 |
| 1 | 10.10.8.2 | 443 | | 0 | 2 | 0 | 4 |
| 2 | 10.10.8.3 | 443 | | 0 | 0 | 0 | 4 |

*Figura 28. Cuadro distribución de usuarios. Fuente: Los Autores*

En la figura 28 se muestra una tabla extraída de la herramienta de balanceo de carga en la que se detalla la distribución de usuarios conectados. Para un total de 12 usuarios, el sistema dividió equitativamente las conexiones entre las 3 máquinas en balanceo, haciendo uso del algoritmo de Round Robin con el que viene incorporado. Esto divide la cantidad de peticiones que recibe cada máquina.

De acuerdo a los datos, se presentaron picos de aproximadamente 70 peticiones por minuto en promedio para cada máquina, en comparación con los picos aproximados 170 peticiones por segundo obtenidos en una prueba sin el sistema de balanceo. Estos fueron recolectados utilizando la herramienta Ganglia.

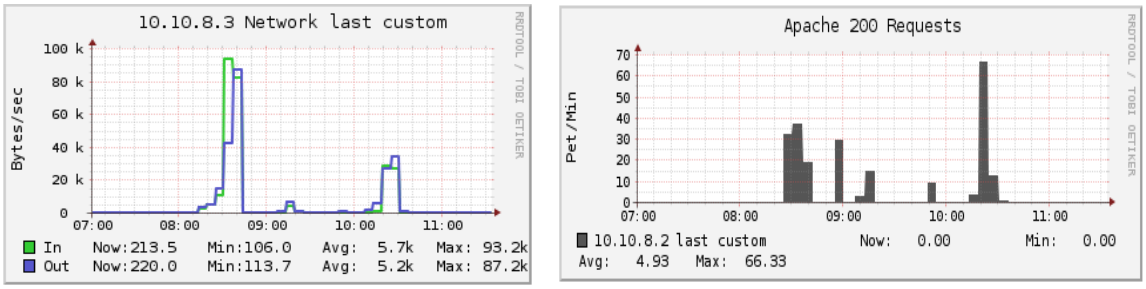



*Figura 29. Carga de red y peticiones sobre una de las máquinas con el sistema de balanceo. Fuente: Los Autores*

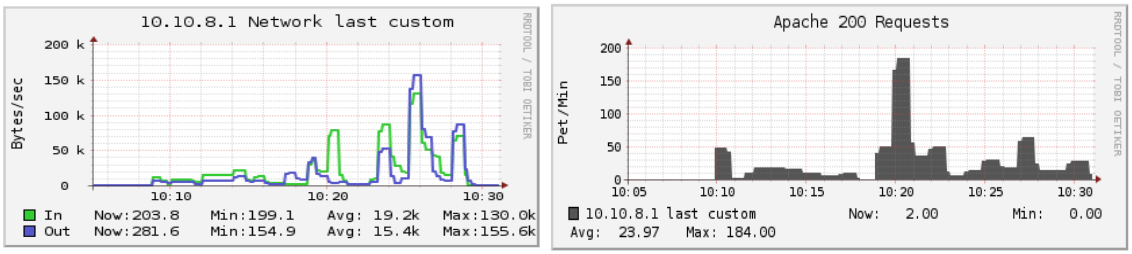



*Figura 30. Carga de red y peticiones sobre una de las máquinas sin el sistema de balanceo. Fuente: Los Autores*

En las figuras se muestra la comparación de la carga de peticiones y de red en escenarios con y sin sistema de balanceo. En la figura 29 se detalla una de las 3 máquinas de la prueba bajo el sistema de balanceo. En la figura 30 se muestran las estadísticas de una prueba sin el sistema de balanceo, en este caso todo el tráfico de red llega a una sola máquina. Se observa una diferencia marcada tanto en el uso de la red como en la cantidad de peticiones que llegan por minuto.

En ítems como CPU y RAM, también se presentaron variaciones, aunque el consumo se mantuvo realmente bajo en ambas pruebas, esto debido a que la asignación de estos recursos fue la suficiente para soportar las aplicaciones de aulas virtuales. Para ver en detalle los resultados obtenidos de la prueba véase el anexo 1.

# 7. CONCLUSIONES

El diseño de infraestructuras de IT con propiedades modulares agrega un grado de flexibilidad que permite la implementación de sistemas ordenados y fácilmente administrables, dejando un margen bastante grande de posibilidades para la migración hacia nuevas tecnologías.

La separación de la funcionalidad en unidades independientes de trabajo y la implementación de sistemas virtuales garantizan un uso más adecuado de los recursos, reduciendo en el proceso la complejidad administrativa y la posibilidad de fallo general de los sistemas.

Los modelos creados asociados con los sistemas de alta disponibilidad y computación en la nube, se convierten en un aporte en el campo de diseño de infraestructuras de IT, permitiendo al grupo CONUSS establecer nuevos proyectos de investigación y siendo puntos de partida para futuros proyectos de la comunidad académica en general.

El uso de tecnologías libres abre las puertas a la creación de sistemas con alto valor agregado y con un nivel de personalización que se acopla perfectamente con los intereses y las estrictas demandas de las organizaciones.

El grupo CONUSS ha sumado esfuerzos sobre varias áreas de investigación, demostrando mediante la ejecución de este y anteriores proyectos, las capacidades que tiene de aportar diseños y productos con alto valor agregado.

Los servicios migrados a entornos virtuales, cuentan con recursos suficientes para atender de forma adecuada las peticiones de usuario, además de un sistema que garantiza su disponibilidad continua y la distribución de la carga asociada, evitando saturación por alta concurrencia.

Con la realización del proyecto se adquirieron conocimientos en el campo de diseño e implementación de infraestructuras de IT que pueden ser fácilmente utilizados en el ámbito científico y profesional.

# 8. RECOMENDACIONES

Crear una interfaz administrativa que permita dar a conocer los servicios soportados por la infraestructura a la comunidad académica, así como realizar un manejo más adecuado de los usuarios que hacen uso de los recursos.

Incorporar tecnologías emergentes como MySQL Cluster, con el fin de brindar un control más especializado de la alta disponibilidad de bases de datos.

Se sugiere la compra de un dispositivo de almacenamiento SAN, para almacenar las imágenes virtuales del sistema y dotar a la infraestructura de un recurso tolerante a fallos que soporte y que asegure los módulos de computación en la nube y de alta disponibilidad.

Con la incorporación de hardware especializado de almacenamiento, utilizar tecnologías de gran soporte y rendimiento como OCFS2 para el manejo de los datos compartidos en red.

Analizar y diseñar un esquema de red basado en la tecnología OpenVSwitch para la creación y separación más adecuada de redes virtuales en el módulo de computación en la nube.

Consolidar el campo de la seguridad informática como una línea de investigación dentro del grupo CONUSS, con el fin de realizar proyectos que ayuden a mejorar la infraestructura actual y aporten nuevos conocimientos a la comunidad académica y científica.

Se sugiere la vinculación del grupo CONUSS a los grupos de investigación de la Universidad Industrial de Santander con el propósito de recibir ayuda y fomento que respalde los trabajos realizados sobre las líneas de investigación elegidas.

Por último se sugiere continuar con la filosofía de uso de tecnologías de software libre en el desarrollo de nuevos proyectos.

# BIBLIOGRAFIA

[1] BARBOSA AYALA, Alexander. MUÑOZ DUARTE, Elkin Dario. Instalación, administración, configuración e implementación de servidores Linux con énfasis en el desarrollo de un modelo administrativo y la creación de un prototipo de clúster de alta disponibilidad. Bucaramanga. Universidad industrial de Santander. Facultad de Ingenierías Fisicomecánicas. Escuela de ingeniería de sistemas e informáticas, 2012.

[2] CABRERA, Sebastián. Virtualización [Webcast]. Colombia: DELL inc. 2012. Sesión 1 (88 min).

[3] CARREÑO DIAZ, Emmanuel. Modelo y prototipo de servicios de computación en la nube para estudiantes y profesores de la escuela de ingeniería de Sistemas e informática de la universidad industrial de Santander. Bucaramanga. Universidad industrial de Santander. Facultad de Ingenierías Fisicomecánicas. Escuela de ingeniería de sistemas e informáticas, 2012.

[4] GEELAN, Jeremy. Twenty one experts define cloud computing. Virtualization, August 2008. Consultado el 15 de Abril de 2013. http://virtualization.sys-con.com/node/612375?page=0,1

[5] GOLDBERG, Robert P. (February 1973) (PDF). Architectural Principles for Virtual Computer Systems. Harvard University. pp. 22–26. Consultado el 15 de Abril de 2013. http://www.dtic.mil/cgi-bin/GetTRDoc?AD=AD772809&Location=U2&%20doc=GetTRDoc.pdf

[6] INSTITUTO COLOMBIANO DE NORMAS TECNICAS Y CERTIFICACION. Tecnología de la información. Técnicas de seguridad. Código de práctica para la

gestión de la seguridad de la información. Bogotá D.C.: ICONTEC, 2007. 133p. NTC-ISO/IEC 27002.

[7] MAGGIANI, Rich. "Cloud computing is changing how we communicate", Professional Communication Conference, 2009. IPCC 2009. IEEE International, vol., no., pp.14, 19-22 July 2009. Consultado el 15 de Abril de 2013. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5208703&isnumber=5208666

[8] MELL, Peter. & GRANCE, Timothy. 2011. The NIST Definition of Cloud Computing. NIST Special Publication 800-145. Consultado el 15 de Abril de 2013. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[9] Microsoft Patterns & Practices Team. Microsoft Application Architecture Guide. Segunda Edición. Estados Unidos: Microsoft Press, 2009.

[10] TASK FORCE ON CLUSTER COMPUTING. High Availability. http://www.cloudbus.org/~raj/tfcc/high-availability.html

[11] VAQUERO Luis M., RODERO-MERINO Luis, CACERES Juan, and LINDNER Maik. 2008. A break in the clouds: towards a cloud definition. SIGCOMM Comput. Commun. Rev. 39, 1 (January 2009), 50-55. Consultado el 15 de Abril de 2013. http://dl.acm.org/citation.cfm?doid=1496091.1496100

# ANEXOS

## Anexo A. Prueba del sistema de balanceo de carga

**1. OBJETIVO:** Validar el funcionamiento del sistema de balanceo de carga incorporado dentro del nuevo modelo así como las aplicaciones migradas a máquinas virtuales y que se apoyan sobre este sistema.

### 2. Características de las máquinas virtuales:

Para el hospedaje de las aplicaciones se dispusieron 3 máquinas virtuales con una partición que implementa un sistema de redundancia, de esta forma cualquier cambio sobre dicha partición en una de las máquinas, se ve reflejado inmediatamente en las demás. Las máquinas se conectan a una base de datos externa necesaria para el funcionamiento adecuado de las aplicaciones. Cada máquina virtual reposa sobre un servidor diferente. A continuación se resumen otras características:

Cantidad: 3

Sistema Operativo: Debian Squeeze 6.0.6

CPU: 1 núcleo de CPU del servidor físico real.

RAM: 1024 MB

Software de balanceo de carga: Zen Load Balancer.

### 3. Descripción de la prueba:

Para la prueba se decidió utilizar un escenario académico, eligiendo un salón de clases perteneciente a la materia Programación Web. Junto con el profesor encargado, se acordaron realizar ciertas actividades para observar el funcionamiento de las aplicaciones hospedadas. En el proceso los estudiantes subieron y descargaron archivos a la plataforma con el fin de medir la concurrencia en el sistema y la carga sobre la red. Este procedimiento se llevó a cabo desde dos escenarios, el primero con el sistema de balanceo de carga habilitado, recibiendo todas las peticiones de los usuarios y distribuyéndolas sobre las máquinas. El segundo escenario planteaba probar la carga sobre una sola máquina deshabilitando el sistema de balanceo. Para la recolección de datos se usó el paquete de monitoreo Ganglia.

**Métricas de medición:**

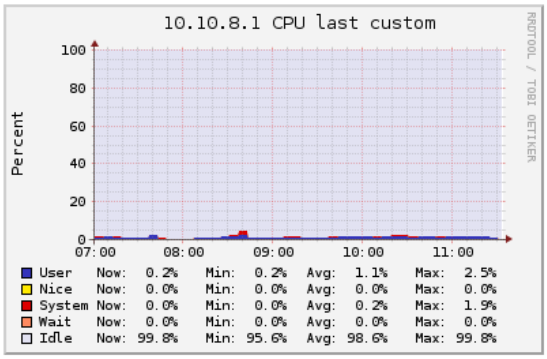
- CPU: Porcentaje de CPU usada
- RAM: Cantidad de RAM usada
- Peticiones web: Número de peticiones por minuto
- Red de datos: Número de Kbps

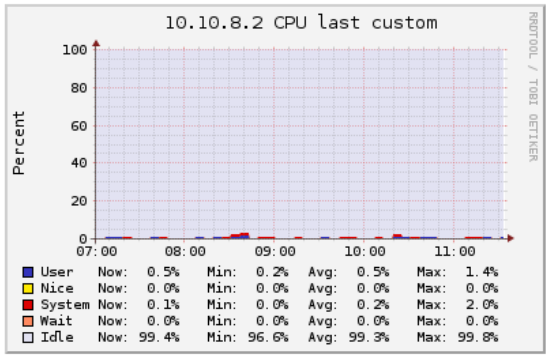
### 4. Realización de la prueba

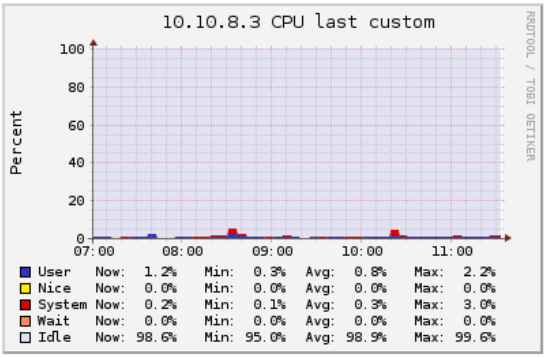
#### 4.1 Escenario 1: Concurrencia bajo el sistema de balanceo de carga

Se habilitó el sistema de balanceo para la distribución de las peticiones entre las máquinas. Los usuarios se conectan a través de una dirección URL sin enterarse de la máquina a la cual son dirigidos. Suben y descargan archivos en un periodo de tiempo establecido. Se tomaron 2 muestras en horas diferentes, una inciando a las 8:20 am y la otra a las 10 am. A continuación se muestran las gráficas comparativas arrojadas por Ganglia.

### Gráficas de uso de CPU

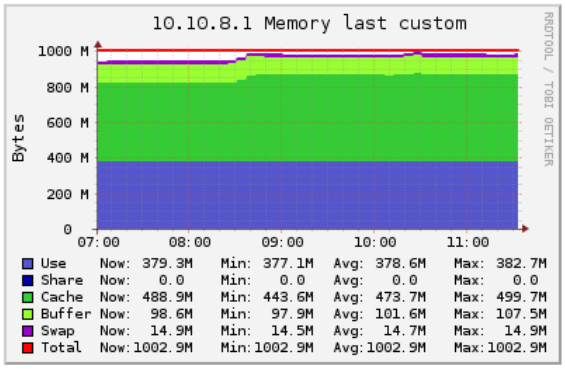


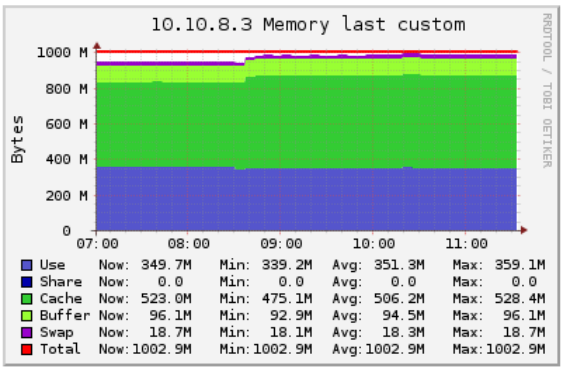



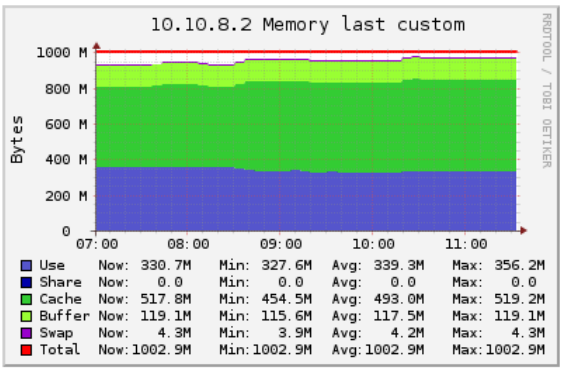




Se observan entonces las variaciones del uso de CPU. Con respecto a las horas de aplicación de las pruebas se presentan variaciones porcentuales mínimas. La concurrencia de usuarios no incrementó significativamente el uso de la CPU en las 3 máquinas.

### Gráficas de uso de memoria RAM

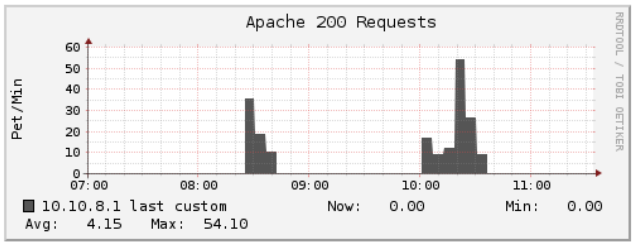


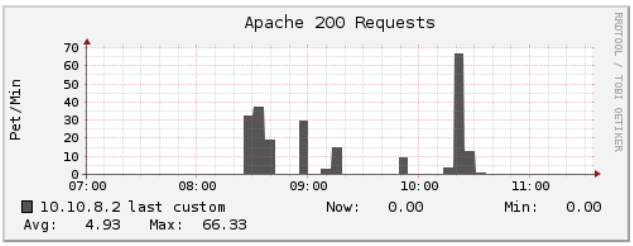



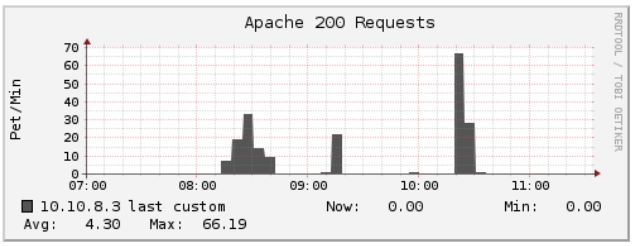



Para el uso de memoria RAM se presenta un caso similar al anterior. En color azul se muestra el uso de este recurso que no presenta cambios grandes en las horas de aplicación de las pruebas.

**Gráficas número de peticiones por minuto**

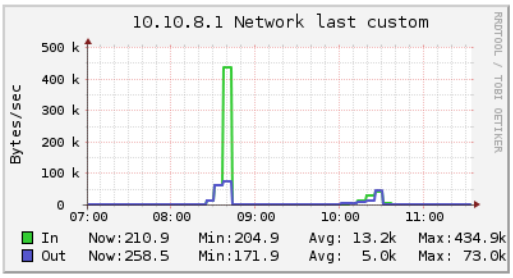


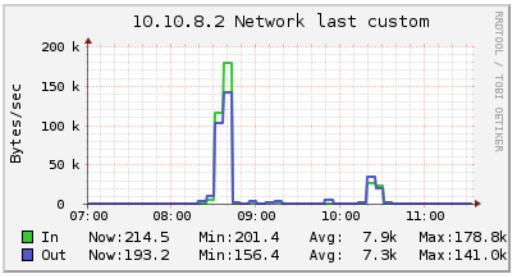



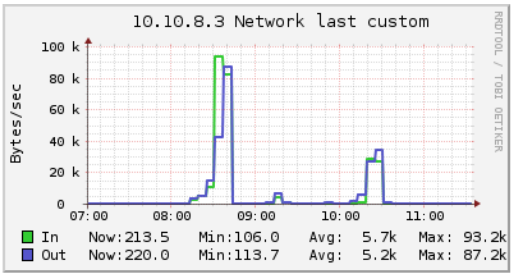



Como se esperaba, se ve incrementado el número de peticiones por minuto desde el momento de aplicación de cada prueba. Esta métrica es muy variable ya que cada usuario sube y descarga distintos archivos en tiempos aleatorios, además hace uso a su manera de los sitios. Aun así, se presentan comportamientos muy similares en las 3 máquinas.

**Gráficas de uso de red**




Estas gráficas, al igual que las anteriores, se presentan valores muy variables dependientes de la forma en que utilicen los sitios web los usuarios. Se observa que las 3 máquinas alcanzan ciertos picos de uso de red según lo esperado por la distribución de peticiones web. El comportamiento es similar a pesar de la variabilidad de las métricas.

**Distribución de usuarios entre las máquinas**

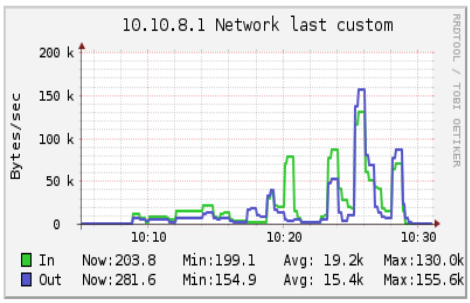
Real servers status 3 servers, 3 current

| Server | Address | Port | Status | Pending Conns | Established Conns | Closed Conns | Clients |
|---|---|---|---|---|---|---|---|
| 0 | 10.10.8.1 | 443 | | 0 | 0 | 0 | 4 |
| 1 | 10.10.8.2 | 443 | | 0 | 2 | 0 | 4 |
| 2 | 10.10.8.3 | 443 | | 0 | 0 | 0 | 4 |

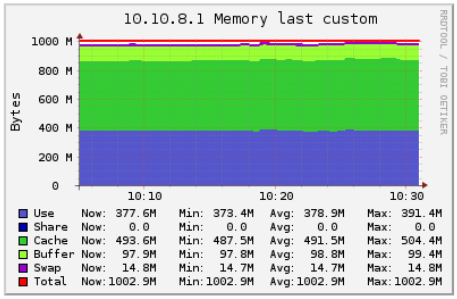
Esta tabla, obtenida desde la herramienta de balanceo de carga, muestra la distribución de usuarios conectados al arreglo de máquinas durante las pruebas. Para un total de 12 usuarios se repartieron equitativamente las conexiones entre los 3 servidores. La herramienta de balanceo cuenta con opciones para mantener vivas las sesiones de usuario por cierto periodo de tiempo, por tanto, los clientes conectados a una de las máquinas permanecían en ella sin que se les cortara la sesión en curso.

### 4.2 Escenario 2: Concurrencia sin el sistema de balanceo de carga

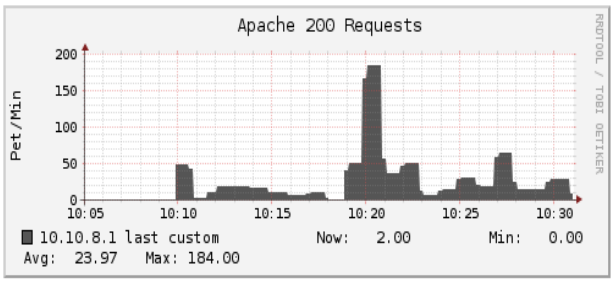
Para este escenario se deshabilitó el sistema de balanceo de carga y se eligió una de las máquinas para atender la totalidad de peticiones de usuario. La prueba se efectuó en un momento diferente con los mismos grupos de estudiantes. En las siguientes gráficas se observa los valores obtenidos para las métricas definidas:



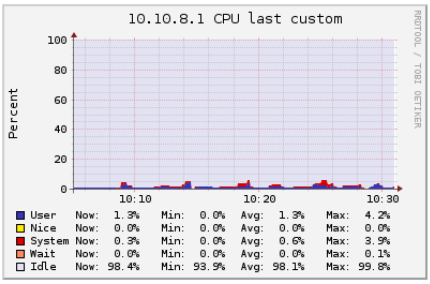

Gráfica: Uso de red




Gráfica: Uso de memoria RAM




Gráfica: Número de peticiones por minuto




Gráfica: Uso de CPU

Las gráficas de CPU y RAM no muestran variaciones importantes. A pesar de soportar la carga ejercida por la totalidad de los usuarios, no se incrementaron significativamente el uso de los recursos. En las gráficas de uso de red y cantidad de peticiones web, se presentó una variabilidad según lo esperado. La cantidad de peticiones se elevaron llegando a picos muy superiores, en comparación con la pruebas con el soporte del balanceador de carga. En cuanto a las métricas de la red, se presenta una cantidad mayor de picos de uso, esto se debe a la cantidad mayor de usuarios que suben y descargan archivos en momentos aleatorios. El ancho de banda no se ve incrementado de manera significativa.