Solución de alta disponibilidad (HA) y balanceo de carga para el Servicio Web de la Red de Datos de la Universidad del Cauca

Jeimmy Viviana Cuellar Rivera

José Raul Romero Mera

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Telecomunicaciones

Grupo I+D Nuevas Tecnologías en Telecomunicaciones

Popayán, Noviembre de 2012

Solución de alta disponibilidad (HA) y balanceo de carga para el Servicio Web de la Red de Datos de la Universidad del Cauca

Trabajo de grado presentado como requisito para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Jeimmy Viviana Cuellar Rivera
José Raul Romero Mera

Director
Ing. Alejandro Toledo Tovar

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
Grupo I+D Nuevas Tecnologías en Telecomunicaciones
Popayán, Noviembre de 2012

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS III
ÍNDICE DE TABLAS III
ÍNDICE DE ECUACIONES IV
ACRÓNIMOS V
INTRODUCCIÓN 1
CAPÍTULO 1: DETERMINACIÓN DE UN CONJUNTO DE CRITERIOS QUE PERMITAN SELECCIONAR Y EVALUAR HERRAMIENTAS LIBRES DE ALTA DISPONIBILIDAD (HA) Y BALANCEO DE CARGA. 4
1.1. CONCEPTOS INICIALES 5
1.1.1. Cluster 5
1.1.2. Tipos de Cluster 6
1.1.3. Configuraciones de redundancia 7
1.2. ALTA DISPONIBILIDAD 8
1.2.1. Tiempo de Caída 8
1.2.2. Causas del Tiempo de caída 9
1.2.3. Métrica de la disponibilidad 9
1.2.4. Definición de alta disponibilidad 10
1.3. BALANCEO DE CARGA 10
1.3.1. Balanceo de carga en servidores 11
1.3.2. Algoritmos de Balanceo de Carga 13
1.4. METODOLOGÍAS DE EVALUACIÓN DE CALIDAD DE SOFTWARE LIBRE 14
1.4.1. Modelos de Calidad de Software Libre de Primera Generación. 16
1.4.2. Modelos de Calidad de Software Libre de Segunda Generación 19
1.4.3. Elección de la Metodología 21
1.5. CRITERIOS DE EVALUACIÓN 28
1.5.1. Definición de criterios para la Fase 1: Evaluación Rápida. 28
1.5.2. Fase 2: Evaluación de Uso Objetivo 30
1.5.3. Establecimiento de los criterios de evaluación de las categorías 32
CAPÍTULO 2: ANÁLISIS DE LA ARQUITECTURA ACTUAL Y PLANTEAMIENTO DEL PROBLEMA 37
2.1. DEFINICIONES GENERALES 37
2.1.1. Aplicación Web 37
2.1.2. Sitio Web con Gestor de Contenidos 38

2.2. ANÁLISIS DE LA ARQUITECTURA ACTUAL PARA SIMCA 39
2.3. ARQUITECTURA ACTUAL PARA EL PORTAL WEB....41
2.4. DISPONIBILIDAD DE LOS SERVICIOS....43
CAPÍTULO 3: DISEÑO Y EVALUACIÓN DE LA SOLUCIÓN....45
3.1. PROYECTOS LIBRES DE ALTA DISPONIBILIDAD Y BALANCEO DE CARGA 45
3.2. EVALUACIÓN RÁPIDA. 47
3.3. DISEÑO DE LA SOLUCIÓN TECNOLÓGICA 51
3.3.1. Diseño de la Solución Tecnológica para el Portal Web 51
3.3.2. Diseño de la Solución Tecnológica para SIMCA 54
3.4. DEFINICIÓN DE TIPOS PRUEBAS Y HERRAMIENTAS DE MONITOREO 56
3.4.1. Pruebas para determinar la funcionalidad 56
3.4.2. Herramientas de monitoreo y de pruebas....57
3.5 ELABORACIÓN DE PRUEBAS Y DETERMINACIÓN DE PUNTAJES PARA LAS DIFERENTES SOLUCIONES TECNOLÓGICAS....58
3.5.1. Elaboración de las pruebas de funcionalidad para la solución tecnológica del portal Web 58
3.5.2. Definición de Puntajes para las soluciones tecnológicas del portal Web 63
3.5.3. Elaboración de las pruebas diseñadas para la solución tecnológica de SIMCA....64
3.5.4. Definición del Puntaje para la solución tecnológicas de SIMCA en su capa de Balanceo de Carga....66
3.6 IMPLEMENTACIÓN EN AMBIENTE REAL....71
3.7 METODOLOGÍA DE SOLUCIÓN....73
3.8 MEJORAS HARDWARE RECOMENDADAS....74
CAPITULO 4: CONCLUSIONES Y TRABAJO FUTURO 76
4.1 CONCLUSIONES 76
4.1 TRABAJO FUTURO 76
Anexo A: Metodología de Evaluación de Calidad OpenBRR.
Anexo B: Proyectos Libres de Alta disponibilidad y Balanceo de Carga.
Anexo C: Soporte de pruebas.
Anexo D: Hojas de Cálculo de OpenBRR.
Anexo E: Manuales de Instalación.
Anexo F: Análisis de Modelos de Evaluación de Calidad de Software Libre.
Anexo G: Informe técnico del Proyecto de Grado.

ÍNDICE DE FIGURAS

Figura 1.1 Cluster en configuración Activo/Pasivo. 7
Figura 1.2 Cluster en configuración Activo/Activo. 7
Figura 1.3 Cluster en configuración N+1 8
Figura 1.4 Cluster en configuración N-to-N 8
Figura 1.5 Balanceo de carga en servidor....12
Figura 1.6 Modelos de Evaluación de Calidad de Software FLOSS 16
Figura 2.1 Esquema General para una aplicación Web. 37
Figura 2.2 Esquema General para un sitio web con gestor de contenidos 38
Figura 2.3 Arquitectura actual para SIMCA. 39
Figura 2.4 Arquitectura actual para el Portal Web. 42
Figura 2.5 Porcentaje de disponibilidad para SIMCA....43
Figura 2.6 Porcentaje de disponibilidad para el Portal Web 43
Figura 3.1 Keepalived + MySQL con replicación 53
Figura 3.2 Keepalived + HA Proxy + MySQL con replicación 54
Figura 3.3 HA Proxy + Keepalived + Pacemaker + MySQL con replicación. 54
Figura 3.4 Arquitectura Física.....55
Figura 3.5 Escenario de Pruebas para la capa de balanceo de la solución tecnológica para SIMCA.65
Figura 3.6 Escenario de Pruebas para la capa de persistencia de la solución tecnológica para SIMCA 67
Figura 3.7 Solución Inicialmente Propuesta 72
Figura 3.9 Solución Modificada 72
Figura 3.10 Metodología de la Solución 73

ÍNDICE DE TABLAS

Tabla 1.1 Métodos, Modelos, Marcos de Referencia y/o enfoques de Evaluación de FLOSS.....15
Tabla 1.2 Asignación de Valores ponderados a los factores OSSM de Navica 18
Tabla 1.3 Marco de comparación de Métodos de Evaluación de Software Libre - *Framework for Comparing Open Source software Evaluation Methods (FOCOSEM)*....22
Tabla 1.4 Comparativo de algunos modelos de evaluación empleando el marco de comparación FOCOSEM 26
Tabla 1.5 Políticas de aprobación. 29
Tabla 1.6 Organización de las categorías según su importancia para el proyecto. 30
Tabla 1.7 Valores de ponderación de las categorías OpenBRR. 32
Tabla 1.8 Criterios para medir la Categoría funcionalidad en proyectos de Alta Disponibilidad y Balanceo de Carga. 33
Tabla 1.9 Criterios de medición para OpenBRR....34
Tabla 2.1 Tiempos de Respuesta Estimados ante un fallo para el portal Web 44
Tabla 2.2 Tiempos de Respuesta Estimados ante un fallo para SIMCA 44

Tabla 3.1 Proyectos enfocados Libres de Alta Disponibilidad y Balanceo de Carga 46
Tabla 3.2 Resultado de la Evaluación Rápida 48
Tabla 3.3 Resultado Fase 1 OpenBRR. 50
Tabla 3.4 Herramientas candidatas para evaluar las siguientes fases 51
Tabla 3.5 Resumen de las pruebas de balanceo de carga para la solución tecnológica Keepalived + MySQL. 59
Tabla 3.6 Tiempo de Caída según el tipo de tráfico para la solución tecnológica Keepalived + MySQL 60
Tabla 3.7 Resumen de las pruebas de balanceo de carga para la solución tecnológica HA Proxy +Keepalived + MySQL. 61
Tabla 3.8 Resumen de los Tiempos de Caída para Keepalived + MySQL. 62
Tabla 3.9 Resumen de las pruebas de balanceo de carga para la solución tecnológica HA Proxy +Keepalived + Pacemaker + MySQL. 62
Tabla 3.10 Resumen de los Tiempos de Caída para HA Proxy+ Keepalived + Pacemaker + MySQL replicación. 63
Tabla 3.11 Puntajes definitivos de Funcionalidad. 63
Tabla 3.12 Resumen de los puntajes obtenidos. 64
Tabla 3.13 Puntajes definitivos para la solución tecnológica 64
Tabla 3.14 Resumen de las pruebas de balanceo de carga para la solución tecnológica para SIMCA 65
Tabla 3.15 Resumen de los puntajes obtenidos 66
Tabla 3.16 Puntaje de funcionalidad para la solución tecnológica dada para SIMCA 66
Tabla 3.17 Puntajes por categoría para los elementos de la solución dada para SIMCA 66
Tabla 3.18 Puntaje total de la solución dada para SIMCA 66
Tabla 3.19 Descripción de la Prueba Nro.1 68
Tabla 3.20 Descripción de la Prueba Nro.2 68
Tabla 3.21 Descripción de la Prueba Nro.3 69
Tabla 3.22 Descripción de la Prueba Nro.4 69
Tabla 3.23 Descripción de la Prueba Nro.1 70
Tabla 3.24 Descripción de la Prueba Nro.2 70
Tabla 3.25 Comparativo de los tiempos de respuesta de elementos para los servicios Web y SIMCA. 71
Tabla 3.26 Comparativo de la disponibilidad actual y lograda para cada elemento. 71
Tabla 3.27 Plan de Hardware para Oracle RAC 11g R2 75

ÍNDICE DE ECUACIONES

Ecuación 1.1 Fórmula básica para el cálculo de disponibilidad. 9
Ecuación 1.2 Fórmula del cálculo de la disponibilidad de los servicios 9

ACRÓNIMOS

AIS*:* *Application Interface Specification* (Especificación de la interfaz de Aplicación).

AST: *Agreed Service Time* (Tiempo Acordado de Servicio).

CDU: Centro de Datos Universitario.

CFE: *Cacheable File Extensions* (Extensiones de archivos almacenables en caché).

CM: *Configuration Management* (Gestión de Configuración).

CMMI: *Capability Maturity Model Integration* (Modelo de Capacidad de Madurez del Software).

CPU: *Central Processing Unit* (Unidad Central de Procesamiento).

CRM*:* *Cluster Resource Manager* (Administrador de recursos del Cluster).

DSN1: *Design Part 1* (Diseño Parte 1).

DT: *Down Time* (Tiempo de caída).

ENV: *Technical Environment* (Medio Ambiente Técnico).

FLOSS*:* *Free/Libre and Open Source Software (Software Libre y de código abierto).*

FOCSE: *Framework for OS Critical Systems Evaluation* (Marco para la Evaluación de Sistemas críticos de sistema operativo).

FOCOSEM: *Framework for Comparing Open Source software Evaluation Methods* (Marco para comparar los métodos de evaluación de software de código abierto).

GPL*:* *General Public License* (Licencia Pública General).

GQM: *Goal-Question-Metrics* (Objetivo de las preguntas de las metricas).

HA: *High Availability* (Alta Disponibilidad).

IP: *Internet Protocol* (Protocolo de Internet).

IPVS: *IP Virtual Server* (Servidor Virtual IP).

ISV: *Independent software vendor* (Proveedores de software independientes).

ITIL: *Information Technology Infrastructure Library* (Biblioteca de Infraestructura de Tecnologías de la Información).

LB: *Load Balancing* (Balanceo de Carga).

LRM: *Local Resource Manager* (Administrador de Recursos Locales).

LVS: *Linux Virtual Server* (Servidor Virtual Linux).

MST: *Maintainability and Stability* (Mantenibilidad y Estabilidad).

MTBF: *Mean Time Between Failure* (Tiempo Medio Entre Fallas).

MTTR: *Mean Time To Repair* (Tiempo Medio de reparación).

O3S: *Open Source Software Selection* (Selección de software de código abierto).

OITOS: *Observatory for Innovation and Technological transfer on Open Source software* (Observatorio de Innovación y Transferencia de tecnología en software).

OMM: *OpenSource Maturity Model* (Modelo de Madurez de Código Abierto).

OpenBQR: *Open Business Quality Rating* (Calificación de Calidad de negocios abiertos).

OpenBRR: *Open Business Readiness Rating* (Calificación de preparación de negocios abiertos).

OSI: *Open Source Initiative* (Iniciativa de Código Abierto).

PDOC: *Product Documentation* (Documentación del Producto).

PMC: *Project Monitoring and Control* (Proyecto de Monitoreo y Control).

PP1: *Project Planning Part 1* (Planificación del Proyecto Parte 1).

PP2: *Project Planning Part 2* (Planificación de Proyectos Parte 2).

PI: *Product Integration* (Integración del producto).

PPQA: *Process and Product Quality Assurance* (Procesos y Aseguramiento de la Calidad del producto).

QSOS: *Qualification and Selection of Open Source Software* (Calificación y Selección de Software de Código Abierto).

QTP: *Quality of Test Plan* (Calidad del Plan de Pruebas)

QualOSS: *QUALity of Open Source Software* (Calidad de Software de Código Abierto).

RAC: *Real Application Clusters* (Cluster de Aplicación Real).

RDMP1: *Availability and Use of a (product) Part 1* (Disponibilidad y Uso de un plan de trabajo (del producto)).

RDMP2: *Availability and Use of a (product) Part 2* (Disponibilidad y Uso de un plan de trabajo (del producto).

REQM: *Requirements Management* (Gestión de Requisitos).

RSKM: *Risk Management* (Gestión de Riesgos).

SI: Sistemas de Información.

SLB: *Server Load Balancing* (Servidor de balanceo de carga).

SIMCA: Sistema Integrado de Matrícula y Control Académico.

SSL: *Secure Socket Layer* (Capa de conexión segura).

SQO-OSS: *Software Quality Observatory for Open Source Software* (Observatorio de Calidad de software de código abierto).

STD: *Use of Established and Widespread Standards* (Utilización de normas establecidas y generalizadas).

STK: *Relationship between Stakeholders* (Relación entre las partes interesadas).

STONITH: *Shoot The Other Node In The Head* (Pégale un Tiro en la Cabeza al otro Nodo).

TI: Tecnologías de la Información.

TST1: *Test Part 1* (Prueba Parte1).

TST2: *Test Part 2* (Prueba Parte 2).

VC: *Vyatta Core* (Núcleo Vyatta).

VCS: *Veritas Cluster Server* (Servidor de Grupo Veritas).

VIP: *Virtual IP* (IP Virtual).

VRRP *Virtual Router Redundancy Protocol* (Protocolo de redundancia de enrutador virtual).

INTRODUCCIÓN

La información se ha posicionado como uno de los principales recursos que poseen las empresas actualmente. Los entes que se encargan de la toma de decisiones han comenzado a comprender que la información no es sólo un subproducto de la conducción empresarial, sino que a la vez alimenta a los negocios y puede ser uno de los tantos factores críticos para la determinación del éxito o fracaso de éstos. Es así como los Sistemas de Información (SI) y las Tecnologías de Información (TI) han cambiado la forma en que operan dichas organizaciones. A través de su uso se logran importantes mejoras, pues automatizan los procesos operativos y suministran una plataforma de información necesaria para la toma de decisiones. Las universidades no son ajenas a este tipo de conceptos, y cada día reconocen más la necesidad que su uso conlleva, creando por ende sistemas que gestionen procesos de calidad, optimicen recursos, o sean puentes de comunicación con la comunidad tanto foránea como interna; todo lo anterior ha constituido a los SI y TI en núcleo fundamental del desarrollo de las actividades cotidianas al punto de volverse dependientes de su existencia y posibilidad de uso.

Bajo este panorama entra a la escena el Centro de Datos Universitario (CDU) de la Universidad del Cauca, como el ente institucional responsable de la administración de los servicios de TI para el beneficio de la comunidad universitaria. Además, la administración del CDU es la responsable de fijar las políticas y estrategias de TI hacia futuro, de tal manera que se aprovechen las ventajas que ofrecen dichos servicios, y con esto sacar el mayor provecho posible de la información existente; sin embargo si se desea maximizar la utilidad que posee la misma, es necesario hacer un manejo correcto y eficiente, tal y cómo se manejan los demás recursos existentes, pero para este fin es necesario comprender de manera general que hay costos asociados con la producción, distribución, seguridad, almacenamiento y recuperación de toda la información que es manejada en la organización. Aunque la información se encuentra alrededor de todos, es imperante comprender que ésta no es gratuita, y que factores tales como el económico se constituyan en un limitante para la calidad de lo que es posible ofertar, viéndose esto reflejado en la actual situación de los servicios brindados en los que la disponibilidad de estos se ha vuelto el talón de Aquiles y en donde se visualiza la urgencia de tomar medidas que permitan hacer frente a este problema.

Es así como surge la propuesta de una solución tecnológica que brinde alta disponibilidad (HA) y balanceo de carga para los servicios Web y SIMCA de la Red de Datos de la Universidad del Cauca, los cuales, se pueden catalogar como críticos, dada la importancia que tienen dentro del desarrollo de las actividades universitarias, buscando así minimizar el impacto que las interrupciones, planificadas o no, generan en el correcto funcionamiento de la institución, esto es posible lograrlo mediante una estrategia eficiente, efectiva y cuya relación costo-beneficio esté acorde a las condiciones actuales.

Adicional a la propuesta se incluye la elección de una metodología para la evaluación de calidad de software libre, la cual surgió dada la necesidad de una elección imparcial y metodológica de herramientas libres. Para el caso específico de este proyecto, se empleó como instrumento de selección y eliminación de herramientas de Alta Disponibilidad y Balanceo de Carga, pero es

importante aclarar que esta también puede ser útil en otros contextos donde la elección de software libre sea necesaria.

Acorde con lo mencionado anteriormente, para el logro de los objetivos propuestos en el anteproyecto, el desarrollo del documento del proyecto se realiza mediante cuatro capítulos, cuyo contenido general se detalla a continuación:

- **Capítulo 1:** En este capítulo se presenta una introducción respecto a los conceptos de alta disponibilidad y Balanceo de carga que son la base importante en el marco del presente proyecto, además se hace una corta presentación de algunas metodologías de evaluación de calidad de software libre, de las cuales se selecciona OpenBRR como la metodología a usar y dentro de su marco de uso se hace la definición de los criterios de evaluación para la solución tecnológica.
- **Capítulo 2:** En este capítulo se presenta un acercamiento hacia los problemas presentados en las actuales arquitecturas para el despliegue de los servicios de Portal Web y el Sistema Integrado para el registro y control académico (SIMCA) así como la presentación de los porcentajes de disponibilidad actuales con que cuentan estos servicios.
- **Capítulo 3:** Este capítulo corresponde al componente práctico del proyecto. Se presentan los diferentes proyectos existentes de uso libre para Alta Disponibilidad y Balanceo de Carga obtenidas tras una exploración tecnológica y mediante la aplicación de las primeras fases de metodología de evaluación OpenBRR se seleccionan algunos de ellos con los que se efectúa la formulación, implementación y pruebas de las potenciales soluciones tecnológicas para validar su funcionalidad y realizar el proceso de selección.
- **Capítulo 4:** Contiene las conclusiones generales con respecto a la realización del trabajo de grado, a los resultados obtenidos. Así mismo, se brinda algunas ideas sobre posibles trabajos futuros.

Para complementar el contenido de los capítulos, se cuenta además con los siguientes anexos:

- **Anexo A: Metodología de Evaluación de Calidad OpenBRR:** Este anexo contiene la descripción detallada de la metodología de evaluación de calidad de Software Libre OpenBRR.
- **Anexo B: Proyectos Libres de Alta disponibilidad y Balanceo de Carga:** Este anexo contiene la descripción detallada de los diferentes proyectos candidatos a ser parte de la solución de alta disponibilidad y balanceo de carga para el Servicio Web y SIMCA de la Universidad del Cauca.
- **Anexo C: Soporte de pruebas:** En este anexo se encuentran todos las capturas realizadas por los diferentes programas de monitoreo y estrés y que sirvieron como base para el proyecto.
- **Anexo D: Hojas de Cálculo de OpenBRR:** Contiene las Hojas de cálculo proporcionadas por la metodología, las cuales se encuentran llenas con los valores correspondientes a las herramientas de la solución tecnológica.
- **Anexo E: Manuales de Instalación:** Este anexo contiene los manuales de instalación y configuración de las herramientas propuestas.
- **Anexo F: Análisis de Modelos de Evaluación de Calidad de Software Libre"**: Artículo científico publicado en modalidad Trabajos en eventos (Capítulos de memoria) Ponencia Internacional

Colombia, Evento: II Encuentro Internacional y VI Nacional de Investigación en Ingeniería de Sistemas e Informática EIISI-2011 Ponencia: Análisis de Modelos de Evaluación de Calidad de Software Libre año: 2011, ISSN: 2248-7948 vol: págs: 95 – 106.

- **Anexo G**: **Informe técnico del Proyecto de Grado**, Este anexo contiene un soporte escrito generado por el ingeniero Jefe del área de Servidores y Servicios de Internet de la División de TIC de la Universidad del Cauca en donde expone su satisfacción con los resultados obtenidos.

CAPÍTULO 1: DETERMINACIÓN DE UN CONJUNTO DE CRITERIOS QUE PERMITAN SELECCIONAR Y EVALUAR HERRAMIENTAS LIBRES DE ALTA DISPONIBILIDAD (HA) Y BALANCEO DE CARGA.

En un mundo que se mueve a velocidades vertiginosas, la necesidad de acceder a los servicios ofertados es cada día mayor, incrementándose los requerimientos de nivel de servicio de los sistemas de información, viéndose todo esto reflejado en la necesidad de arquitecturas tolerantes a eventualidades (catástrofes naturales, errores humanos, fallos del sistema, etc.) [1], máxime cuando los mismos representan parte vital del desarrollo de las actividades de una institución y cuando su falla puede constituirse en un problema que genera costos directos e indirectos, derivando en factores que van desde entorpecimiento de las labores hasta cese completo de las mismas, es así que la alta disponibilidad se hace un paradigma necesario y vital para un centro de datos que maneja servicios críticos. Esto da como resultado que cada vez más empresas y universidades se vean avocadas a la implementación de técnicas que permitan a los servicios estar disponibles todo el tiempo para su uso.

Especialmente para la Universidad del Cauca, se hace necesario el planteamiento de una solución tecnológica que permita la disponibilidad de los servicios Web y Sistema Integrado de Matrícula y Control Académico (SIMCA) ofertados por la Red de Datos[1], dichos servicios no sólo fueron elegidos por los altos niveles de tráfico que presentan sino también por la marcada importancia que tienen dentro del ámbito universitario. El primero de ellos, tiene su importancia al ser la cara visible de la universidad ante el mundo, donde además se encuentra la información de comunicados, resoluciones, acuerdos, circulares, eventos, información general, entre otros, constituyéndose así en un importante puente de comunicación e información tanto para la comunidad universitaria como la foránea; el segundo de ellos, SIMCA, es un elemento vital de planeación académica, en donde es posible hallar información sobre: espacios físicos, docentes temporales, catedráticos y de planta, movilidad inteligente, proceso de matrícula, registro y control, egresados y labor docente [2], elementos que resultan significativos para el correcto desarrollo de las actividades universitarias. Dada la vital labor que desempeñan estos dos elementos y el alto índice de visitas que reciben, se hace necesaria la implementación de técnicas que aseguren su disponibilidad el mayor tiempo posible así como de aquellas técnicas que puedan ayudar a elevar la eficiencia en el uso de los recursos con que se cuenta, con el fin de aumentar el tiempo en que estas plataformas están disponibles para su uso y el número de usuarios que pueden acceder en un mismo espacio de tiempo; sin embargo se debe tener en cuenta que mejorar la disponibilidad no es necesariamente un equivalente de rentabilidad [4], por lo tanto, la solución tecnológica a brindar debe ser un balance entre el costo y el beneficio a obtener.

[1] La Red de Datos de la Universidad del Cauca, es el nombre con el que se conoce o identifica a las instalaciones físicas donde se encuentran los equipos, infraestructura y personal necesario para desarrollar las funciones que permiten la prestación de servicios de Tecnologías de la Información (TI) entre otros.
Aunque actualmente persiste el nombre de Red de Datos, y es con el que se identifica, ya no corresponde, ahora existe la División Gestión de las Tecnologías de Información y Comunicación, una dependencia que hace parte de la Vicerrectoría Administrativa y está encargada de promover el desarrollo tecnológico de la Universidad del Cauca, bajo la cual se agrupan cuatro áreas: Servidores y Servicios, Infraestructura, Help Desk y Desarrollo. Dentro de estas áreas, la correspondiente a la Red de Datos, es el Área de Servidores y Servicios, que tiene a su cargo la administración y soporte de los servicios [3].
A fin de tener coherencia con el anteproyecto de grado y el título del proyecto se seguirá con la denominación Red de Datos, teniendo en cuenta la aclaración anterior.

Es por lo anterior que se encontró que la solución tecnológica debe ser basada en software libre y el hecho es que cuando se habla de software libre se hace necesario entender que este cuenta con características que son intrínsecas al mismo, las cuales lo hacen diferente al software de tipo privativo y antes de considerar tenerlo en una empresa es necesario hacerse preguntas como: *"¿Cuál es la durabilidad del software?, ¿Cuáles son los riesgos de Forks[2]? ¿Cómo anticipar y gestionarlos?, ¿Qué nivel de estabilidad esperar?, ¿Cuál es el nivel de soporte disponible?, ¿Es posible influir en un mayor desarrollo del software (adición de nuevas funcionalidades específicas)?"* [5]; todas estas preguntas deben ser contestadas antes de la inclusión del mismo. Es así como el desafío que presenta el proyecto es el mismo que se presenta al interior de muchas compañías, la elección de una determinada solución tecnológica software que cumpla con un requerimiento o necesidad particular [6], pero se hace aún más complejo en el caso del software libre dados los mitos existentes en torno a él y a la dinámica en su desarrollo el cual gira alrededor de una comunidad de desarrolladores y usuarios, por esto se hace particularmente importante examinar con precisión las limitaciones y los riesgos específicos de software libre y/o de código abierto, por lo que es imprescindible contar con un método que permita hacer una evaluación cualitativa mediante la integración de las características de código abierto y una comparación según los requisitos de las necesidades formales de criterios ponderados, con el fin de tomar una decisión final, la cual debe permitir seleccionar alguna solución tecnológica que satisfaga requisitos funcionales, estratégicos y técnicos.

En este capítulo, inicialmente se mostrará los conceptos de alta disponibilidad y Balanceo de carga, definiendo un marco teórico formal que aporte los conceptos necesarios para el proyecto y posibilite el desarrollo de una propuesta de solución tecnológica de Alta Disponibilidad y Balanceo de carga; en segundo lugar, se realizará una exploración de las diferentes metodologías de evaluación de calidad de software libre, eligiéndose aquella que cuente con los elementos necesarios para su uso dentro del presente proyecto y para con su uso determinar el conjunto de criterios que permitan seleccionar y evaluar herramientas libres de Alta Disponibilidad y Balanceo de Carga.

1.1. CONCEPTOS INICIALES

1.1.1. Cluster

Es una palabra inglesa cuya traducción literal es "racimo" o "grupo", y es usada para hacer referencia a un conjunto de entes que realizan un trabajo conjunto para lograr un fin específico [7]; en el caso de la informática, un *Cluster* de computadores se basa en una arquitectura que comprende componentes de *hardware* y *software*, interconectados mediante tecnologías de red de alta velocidad y configurados de forma coordinada a fin de dar la ilusión de un único recurso; cada uno de estos sistemas estará proveyendo un mismo servicio o ejecutando una (o parte de una) misma aplicación paralela. Un *cluster* debe tener como característica inherente la compartición de recursos: ciclos de CPU (*Central Processing Unit*), memoria, datos y servicios.

[2] Literalmente traduce bifurcación, en software se usa para identificar la creación de proyectos nuevos a partir de otro ya existente.

Los dispositivos computacionales (nodos) que conforman un *cluster* pueden tener todos la misma configuración de hardware y sistema operativo (*cluster* homogéneo), diferente rendimiento pero con arquitecturas y sistemas operativos similares (*cluster* semihomogéneo), o tener diferente hardware y sistema operativo (*cluster* heterogéneo); estos nodos a su vez pueden hallarse en un *rack* dentro de un espacio dedicado exclusivamente a almacenar computadores, o pueden estar ubicados en una oficina pequeña dentro del cubículo de un empleado; lo que cuenta realmente es como están relacionados, como son accedidos y el tipo de aplicación que están ejecutando [8], teniendo en cuenta que para que esto se dé y los diferentes nodos actúen como si se tratara de un solo servidor, los archivos deben encontrarse almacenados de tal forma que puedan ser accedidos por todos los nodos del *cluster*.

Se pretende que la administración de un *cluster* sea lo más sencilla posible considerándose como deseable el hecho de que si bien no pueda ser igual a administrar un solo nodo, por lo menos llegue a ser lo más parecido posible, esperándose por tanto que su software de administración se encuentre en la capacidad de proveer dicho nivel de transparencia [9] [10] [11] [12].

1.1.2. Tipos de Cluster

Existen diferentes tipos y usos que se siguen considerando formas de *cluster* las cuales son:

A. ***Cluster* Alto rendimiento (*High Performance computing*):** Este tipo de cluster resulta atractivo para las industrias, la comunidad o cualquier estamento donde sea necesaria la resolución de problemas complejos o simulaciones que demanden un alto uso de recursos computacionales, aun así su uso se restringe a la ejecución de programas paralelizables[3] ya que este tipo de *cluster* realiza una distribución de las diferentes tareas entrantes hacia un conjunto de equipos que las procesa, logrando de esta manera que su capacidad de procesamiento sea equivalente a la sumatoria de la capacidad individual de todos los miembros que le componen [13].

B. ***Cluster* de Alta disponibilidad (*HA, High Availability*):** Son *Clusters* diseñados con la finalidad de lograr disponibilidad y confiabilidad en el servicio, esto se logra a nivel hardware mediante la eliminación de puntos únicos de fallo y a nivel software mediante la instalación de herramientas capaces de detectar fallos y hacer la recuperación de los mismos migrando la carga hacía un nodo de respaldo [12][14].

C. ***Cluster* de Balanceo de Carga (*LB, Load Balancing*):** Su finalidad es hacer una distribución de la carga o peticiones entrantes a un conjunto de equipos adscritos al mismo y que son los encargados de su procesamiento. Es utilizado principalmente en servicios de red sin estado, como por ejemplo un servidor web con un alto tráfico de red y carga de trabajo [13].

Es importante aclarar que un mismo *cluster* puede ser una mezcla de los tres tipos. Para el presente proyecto se hace necesaria la combinación de dos de los tres tipos, haciendo uso de los *clusters* de balanceo de carga y alta disponibilidad.

[3] *"Cómputo paralelo es la división de una gran tarea o trabajo computacional de cierta cantidad de partes para ejecutarse de manera simultánea en todos los nodos. La gran mayoría de las tareas son fraccionables."* [9]

1.1.3. Configuraciones de redundancia

Es posible encontrar diferentes tipos de configuraciones de redundancia, las cuales se encuentran referidas a la distribución de recursos y la función específica desempeñada por cada nodo, algunas de estas son:

A. Activo/Pasivo: En este tipo de configuración se encuentra el modelo denominado "Maestro-Esclavo" (Figura 1.1), teniéndose así un nodo que actúa como maestro o nodo primario el cual tiene desplegado los recursos y el nodo esclavo o secundario que contiene el servicio a desplegar pero únicamente entra en funcionamiento cuando se presenta un fallo en el primario [15].

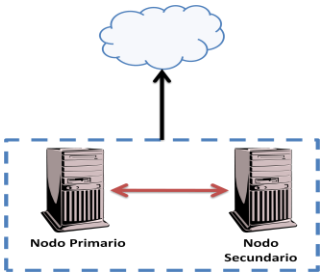

Figura 1.1 Cluster en configuración Activo/Pasivo.

B. Activo/Activo: *Cluster* donde los dos nodos actúan como servidor primario desplegando ambos el servicio (Figura 1.2), quedando así habilitados para responder a las peticiones entrantes en "igualdad" de condiciones, su principal uso es en el balanceo de carga ya que se presenta una división entre las peticiones realizadas por los usuarios, siendo para ellos, un proceso transparente.

En caso de presentarse un fallo en uno de los dos nodos este es retirado del *cluster* y la carga destinada al mismo será asumida por el nodo que aún se encuentra en funcionamiento, siendo este también un proceso transparente para el cliente [15].

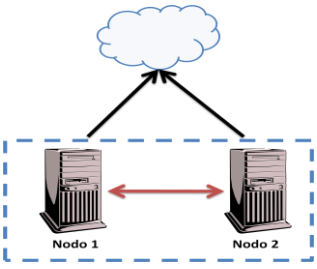

Figura 1.2 Cluster en configuración Activo/Activo.

Es posible también hallar otro tipo de configuraciones para la construcción de *clusters* de alta disponibilidad, las cuales son de carácter más avanzado, estas son:

C. N+1: Al utilizar una configuración de tipo activo-pasivo puede resultar demasiado costoso la implementación de un respaldo para cada nodo, máxime si se tienen en cuenta que estos no entran en operación a menos que se presente un fallo, resultando en algunos casos en una subutilización de los recursos, así que a fin de reducir los costos de un nodo de respaldo que no hace nada, se pueden diseñar configuraciones en las que el nodo de secundario o de

respaldo sea compartido por varios nodos principales (Figura 1.3), obteniéndose así un sistema de alta disponibilidad, ya que aún se cuenta con tolerancia a fallo en los nodos primarios [15].

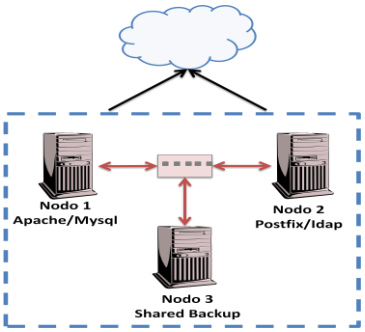

Figura 1.3 Cluster en configuración N+1

D. N-to-N: En esta clase de C*luster* cada nodo puede potencialmente ser utilizado para la conmutación por error, ya que cada nodo puede contener cualquier servicio, pero para esto es necesario contar con almacenamiento compartido [15] (Figura 1.4).

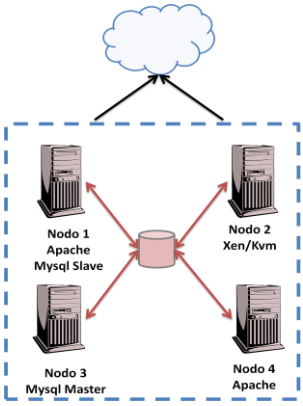

Figura 1.4 Cluster en configuración N-to-N

1.2. ALTA DISPONIBILIDAD

La disponibilidad de manera general se puede definir como el tiempo que un sistema, aplicación o servicio es capaz de realizar las funciones para las que está diseñado, este tiempo puede ser pactado y puede ir desde las 24 horas los siete días de la semana hasta los rangos de tiempo específico [16] [17].

1.2.1. Tiempo de Caída

En cualquier circunstancia donde un sistema no esté en la capacidad de realizar las funciones para las que está diseñado, sin importar las causas que pudieron conducir a esta situación, se considera que el mismo esta "abajo"; *"Por lo que se dice que el sistema sufre de un tiempo de caída, entonces tiempo de caída se define como: el tiempo que los usuarios tienen que esperar para que el sistema se encuentre funcionando nuevamente"* [17].

1.2.2. Causas del Tiempo de caída

Los tiempos de caída pueden ser planeados o no planeados. Los tiempos de caídas planeados son aquellos que se realizan de forma programada, y su finalidad es la realización de mantenimientos preventivos, actualizaciones o mejoras en el sistema, generalmente se eligen las horas en las que presente menor traumatismo a las actividades, siendo típica su realización durante las horas de menor afluencia de usuarios al sistema, se busca en lo posible, minimizar los mismos siendo deseable realizar dichas actividades manteniendo el sistema habilitado para su uso. El tiempo de caída no planeado es aquel que afecta el sistema de forma inesperada, siendo el resultado de múltiples causas entre las que es posible encontrar fallas en componentes tanto hardware como software por: desastres naturales, defectos de fabricación o programación entre otros, obligando por tanto a tomar medidas de tipo correctivo para lograr el correcto funcionamiento del sistema [17] [18] [19].

1.2.3. Métrica de la disponibilidad

La disponibilidad puede ser calculada mediante la ecuación básica (Ecuación 1.1)

$$\boldsymbol{A = \frac{MTBF}{MTBF + MTTR}}$$

Ecuación 1.1 Fórmula básica para el cálculo de disponibilidad.

Donde A (*Availability*) es el grado de disponibilidad expresado como porcentaje, *MTBF (Mean Time Between Failure)* es el tiempo medio entre fallas y *MTTR (Mean Time To Repair)* es el máximo tiempo para la reparación una vez ocurrida la falla.

Otro modo de medir la disponibilidad es el tomado de ITIL (*ITInfrastructure Library*) (Ecuación 1.2), este relaciona el tiempo de acuerdo del servicio respecto al tiempo de interrupción del mismo, esta fórmula puede resultar más práctica ya que es más fácil conocer un tiempo acordado de servicio y el tiempo de interrupción del mismo, en vez del tiempo entre acciones correctivas o de mantenimiento, que resulta más dispendioso y menos familiar [18]. La fórmula según [20], es la siguiente:

$$\boldsymbol{\%Disponibilidad = \left(\frac{AST - DT}{AST}\right) * 100}$$

Ecuación 1.2 Fórmula del cálculo de la disponibilidad de los servicios

Donde %*Disponibilidad* es el grado de disponibilidad expresado como porcentaje, AST (*Agreed Service Time*) es el tiempo acordado en que el sistema debe estar funcionando y DT (*Down Time)*, es el tiempo en que el sistema se encuentra interrumpido o es inutilizable durante las franjas horarias establecidas para su uso.

En esta última fórmula existen dos variaciones para establecer el tiempo de indisponibilidad, una es considerando el sistema como un único componente y la otra es considerándolo como componentes separados, en la primera al asignar el tiempo de disponibilidad se hace de manera global y es independiente de cuantos componentes tenga el sistema, la segunda es asignar una disponibilidad a cada componente y sumar por separado el tiempo de indisponibilidad de cada

uno, un ejemplo de esto es un sistema que funciona 24 horas al día, los 365 días del año y que cuenta con 7 componentes, cada uno con un porcentaje de disponibilidad del 99.99%, así que cada componente tendrá un tiempo de inactividad o indisponibilidad de 52 minutos (sujeto a disponibilidad del 99.99 %), asumiendo que los 7 componentes no fallen al mismo tiempo se tiene: 7 por 52 minutos, 364 minutos, o poco más de 6 horas al año, lo cual generaría una disponibilidad del 99,93% al año [19]; dado que definir la disponibilidad de componente en un centro de datos universitario es engorroso y por demás ineficiente, los datos globales presentados en el desarrollo del presente trabajo de grado se realizaran asignando al sistema una disponibilidad total, pero para efectos comparativos se utilizó los tiempos medios de reparación para cada componente.

1.2.4. Definición de alta disponibilidad

"*La Alta Disponibilidad es la característica que tiene un sistema para protegerse o recuperarse de interrupciones o caídas, de forma automática y en un corto plazo de tiempo*" [21], Estrictamente hablando, el término alta disponibilidad se relaciona con un porcentaje de disponibilidad dentro del rango 99%-99.999%, siendo habitual la referencia del nivel máximo, conocido como los cinco nueves de disponibilidad [22].

La alta disponibilidad en el marco de este trabajo de grado será abordada, desde la necesidad específica de un centro de datos universitario, y se tomará como la capacidad que tiene el mismo para brindar acceso a los recursos que posee, con un mínimo de interrupciones programadas y no programadas, y en caso de que éstas se produzcan, el tiempo de recuperación del servicio debe ser mínimo, mitigando así el impacto del tiempo de inactividad generado [18]**,** así que a pesar de que los valores mencionados serían deseables hay que tener en cuenta que son muy difíciles de lograr incluso para centros de datos de grandes empresas, los cuales cuentan con suficientes recursos económicos, dado el alto costo que implica involucrar integralmente todos los aspectos de un centro de datos, como la infraestructura tecnológica y física, el recurso humano y los recursos económicos que sustentan todo y por tanto fijan el límite del alcance de una solución tecnológica de alta disponibilidad. Además dicho intervalo de disponibilidad está justificado por un enfoque exclusivamente comercial y no son aplicables o siquiera cruciales para el desarrollo de este trabajo.

1.3. BALANCEO DE CARGA

Para solucionar el problema de sobrecarga en los servidores, se empleaba con frecuencia el DNS para distribuir la carga a través de varios servidores, mediante la implementación de una de las configuraciones de direcciones IP en la forma Round-Robin para cada consulta DNS; la desventaja de esta forma es que no se tenía en cuenta el estado del servidor ni su carga porque el DNS fue inventado con el propósito de proveer un sistema de traducción de nombre a dirección para Internet y no para efectuar balanceo de carga [23] [24], es por esto que el balanceador de carga surge como una solución, al incorporar mecanismos de revisión periódica de cada uno de los servidores, de tal forma que permite excluir aquellos que están caídos o con su capacidad excedida de una manera transparente para los usuarios.

El balanceo de carga se le define como el puente existente entre los servidores y la red [25], igualmente se concibe como una técnica bien establecida para la utilización efectiva de recursos

de computación disponibles mediante la partición de tareas de acuerdo a estrategias de distribución de cargas [26] ó la práctica de distribución de trabajo con uniformidad entre múltiples equipos de cómputo [27] que provee muchos beneficios importantes como [28]:

- Reducción de carga en un sistema individual y minimización de tiempos de respuestas.
- Incremento en la utilización de la red y maximización del *throughput.*
- Mejoramiento de la confiabilidad.
- Facilidad de escalamiento.
- Mejoramiento en la satisfacción del usuario.

Además de sus múltiples beneficios es de destacar que los balanceadores de carga pueden ser usados con muchas aplicaciones diferentes, aunque su uso más generalizado sea la gestión de servidores Web, no obstante sus cuatro contextos principales de aplicación son:

A. El balanceo de carga en servidores: Divide la cantidad de trabajo que un servidor debe realizar distribuyendo la carga entre dos o más servidores, por lo que pueden procesar más trabajo en el mismo periodo de tiempo, haciendo que se escale la capacidad y se tolere mejor las fallas en uno de los servidores [29].

B. Balanceo de servidor Global: Se usa cuando un sitio web o los servicios están alojados en distintos servidores situados en diferentes localizaciones geográficas. Con esta tecnología, el tráfico de Internet puede ser distribuido entre los diferentes centros de datos ubicados en diferentes lugares del planeta. Esta tecnología es altamente eficiente en evitar tiempos de caída [30].

C. Balanceo de carga de *Firewall*: *"Uno de los típicos "puntos únicos de fallo" de las topologías de red de los servicios web suele ser el servidor que conecta la red con el exterior, normalmente el firewall"* con balanceo de carga lo que se hace es distribuir la carga mediante múltiples firewalls para con esto escalar más allá de la capacidad de uno sólo y además tolerar una falla al detectarla y retirar el nodo involucrado [31].

D. Conmutación de cache transparente: dirige el tráfico en forma transparente a caches para acelerar el tiempo de respuesta para los clientes o mejorar el desempeño de servidores web por fuera de carga, descarga el contenido estático para caches.

Dado que tanto el Balanceo de servidor Global, como el Balanceo de carga de *Firewall,* y la Conmutación de cache se encuentran fuera del alcance de aplicación del presente proyecto, se desarrollará a continuación únicamente el correspondiente a balanceo de carga en servidores.

1.3.1. Balanceo de carga en servidores

Los balanceadores de carga en servidores [32] (*Server Load Balancing – SLB*) han emergido como una solución potente para aplicaciones corrientes y direccionan diversas áreas, incluyendo escalabilidad en granja de servidores, disponibilidad, seguridad y capacidad de gestión [28].

Debido a que Internet consiste en un número de usuarios quienes solicitan un servicio particular, cada usuario es identificado por una dirección IP, lo cual facilita distribuir la carga a través de múltiples servidores que proveen el mismo servicio o ejecución de la misma aplicación.

El balanceador de carga se coloca al frente de una granja de servidores llamados servidores reales, estos pueden estar interconectados directamente o mediante un *switch* al balanceador de carga y desde la perspectiva o punto de vista de los usuarios, conforman un servidor virtual. El servidor virtual debe tener asignada una dirección IP para que los usuarios puedan accederlo, esta dirección IP se denomina dirección IP virtual o *Virtual IP* (VIP) y se configura en el balanceador de carga para representar la granja de servidores reales completa (Figura 1.5).

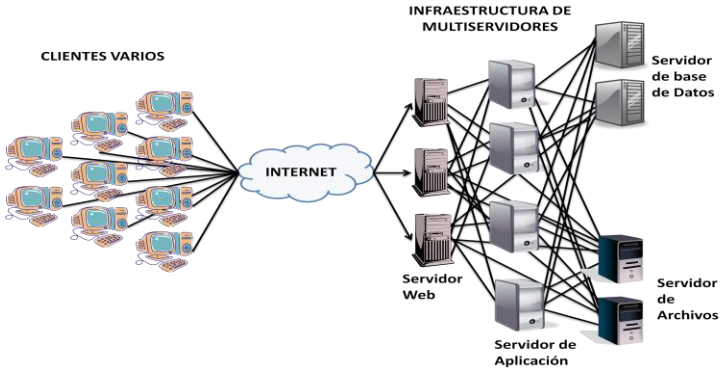

Figura 1.5 Balanceo de carga en servidor [32]

Si el DNS autoritativo está configurado para entregar la VIP como la dirección IP del sitio, todos los navegadores de los clientes enviaran solicitudes a la dirección VIP para que el balanceador de carga reciba las peticiones y las distribuya a través de los servidores reales disponibles.

Los beneficios que se obtienen al implementar una solución tecnológica de balanceo de carga son:

A. Escalabilidad: Al repartir o distribuir las solicitudes de los clientes mediante el uso de algoritmos de distribución de carga entre múltiples servidores reales disponibles, si el algoritmo es perfecto, la capacidad del servidor virtual sería igual a la capacidad agregada de todos los servidores reales. Pero esto rara vez se da, sin embargo, aun si la capacidad del servidor virtual es equivalente al 80 o 90% de la sumatoria de las capacidades individuales de procesamiento de todos los servidores reales disponibles, esto resulta mucho más económico que la implementación de un sólo servidor real que iguale a la capacidad combinada de procesamiento.

B. Disponibilidad: Con un monitoreo continuo del estado de los servidores reales y las aplicaciones que corren en ellos, el balanceador de carga puede detectar si en uno de estos presenta algún tipo de inconveniente o falla, en el momento en que esta situación se presenta, las conexiones existentes y solicitudes que estén en procesamiento se pierden y el balanceador de carga direccionará todas las solicitudes nuevas a uno o varios de los servidores reales que tenga buen estado. Debido a que el servidor hace esto en forma transparente al usuario, el tiempo de caída se minimiza. Una vez que el servidor con falla ha sido reparado, el balanceador de carga detecta el cambio de estado e inicia el reenvío de solicitudes al servidor.

C. Capacidad de gestión: Permite que las labores de mantenimiento por parte de los administradores del centro de datos se efectué sin afectar el servicio para los usuarios. Si se requiere realizar una actualización de hardware o software en el servidor, el mismo debería "bajarse", esto podría programarse en horas de menor tráfico, buscando con esto minimizar el impacto generado, aun así se presentará un *downtime* o tiempo de caída, pero algunos negocios serían incapaces de encontrar el tiempo para realizar este tipo de operaciones, en especial si el servidor es accedido por usuarios alrededor del globo en husos horarios diferentes. Con el despliegue de un balanceo de carga, se puede colocar el servidor fuera de línea en forma transparente para mantenimiento sin requerirse un tiempo de caída, esto se logra desempeñando un apagado automatizado del nodo involucrado para lo cual se suspende la entrega de nuevas solicitudes al mismo y se espera hasta que las conexiones existentes terminen, haciendo que las nuevas peticiones destinadas a la VIP sean distribuidas a través del resto de servidores reales.

Los balanceadores de carga, también ayudan con la gestión de gran cantidad de contenido, conocido como gestión de contenido. Esto es mediante la organización de servidores en diferentes grupos, cada grupo es responsable para una cierta parte del contenido y el balanceador de carga dirige las peticiones al grupo apropiado basado en los URL de las peticiones HTTP. Los balanceadores de carga pueden distribuir la carga a cualquier servidor sin distinción alguna de su sistema operativo, esto permite a los administradores mezclar y juntar diferentes servidores y aun tomar ventaja de cada servidor para escalar la capacidad de procesamiento agregada.

D. Seguridad: Debido a que los balanceadores de carga son el "*front end*" de los servidores de granja, pueden proteger a los servidores de usuarios maliciosos; muchos productos de balanceo de carga viene con características de seguridad para ciertos tipos de ataques. El balanceador de carga puede naturalmente ser un intermediario que desempeña traslación de direcciones de red como parte de la distribución y reenvío de peticiones a diferentes servidores reales. El VIP en el balanceador de carga puede ser una dirección IP pública para que los usuarios en Internet puedan acceder a la VIP, pero los servidores reales detrás del balanceador de carga pueden tener direcciones IP privadas para forzar la comunicación a ir a través del balanceador de carga.

E. Calidad de servicio: Puede ser definido en muchas formas diferentes, como: respuesta en tiempo de un servidor o aplicación, disponibilidad de un determinado servicio aplicativo, o la capacidad de proveer servicios diferenciados basados en el tipo de usuarios. Los balanceadores de carga pueden ser usados para distinguir los usuarios basado en alguna información en la petición de paquetes, y dirigirlas al servidor o grupo de servidores, o configurar los bits de prioridad en el paquete IP para proveer la clase de servicio deseado.

1.3.2. Algoritmos de Balanceo de Carga

Los algoritmos o métodos de Balanceo de Carga son empleados para decidir cual servidor del grupo de servidores o granja de servidores podría ser asignado a la nueva conexión solicitada. Existen varios algoritmos que pueden ser usados, alguno de ellos son:

A. Balanceo Uniforme (*Uniform Balancing*).

B. Balanceo ponderado uniforme (*Uniform Weighted Balancing*).

C. *Simple Round Robin.*

D. *Weighted Round Robin.*

E. Servidor de Tiempo de Respuesta (*Server Response Time*).

F. *Simple Least Connections.*

G. *Weighted Least Connections*.

H. Conexiones Máximas (*Maximum Connections*).

Hasta este punto se ha explorado los conceptos de alta disponibilidad y balanceo de carga, y mediante la apropiación de estos conceptos se hace posible tener claro no sólo su significado, sino además su función, ámbito de aplicación, requisitos técnicos y resultados esperados con su aplicación, lo cual resultará útil al desarrollar los criterios propios que permitan la evaluación de las posibles soluciones. Sin embargo, dado lo extenso del tema no es posible tratar en su totalidad a profundidad todos ellos, lo cual se deja para estudios en sus respectivas referencias. A continuación, se presenta brevemente algunos conceptos sobre el Software Libre y metodologías de elección existentes.

1.4. METODOLOGÍAS DE EVALUACIÓN DE CALIDAD DE SOFTWARE LIBRE

El término "FLOSS" (Free/Libre and Open Source Software [33]) es utilizado para referirse tanto al Software Libre, (*Free Software*, definido por la *Free Software Foundation* [34]*)*, como también al Software de Código Abierto, (*Open Source Software*, definido por la *Open Source Initiative* [35]). De igual forma, FLOSS se define como las acciones que una persona puede realizar en una parte de software que ha recibido y que son respaldadas con los permisos otorgados por el titular de los derechos de autor mediante una licencia de distribución debidamente reconocida por *Free Software Foundation* y/o *Open Source Initiative* [36]. En el caso de las licencias del Software Libre, los usuarios tienen la libertad de ejecutar el programa para cualquier propósito, de estudiar y modificar el programa, y de distribuir copias, ya sea del programa original o modificado, gratis o por cobro de una tarifa por distribución [34]. "*El disponer del código fuente no implica la posibilidad de distribuir copias modificadas o la libertad de adaptarlo a nuestras necesidades (que sí posee el software libre)"* [37], esto puede ser restringido según el tipo de licencia que tenga el producto FLOSS [38].

En los últimos años, los proyectos FLOSS han ganado terreno en el mercado y muchas compañías muestran interés a pesar que aún se mantienen ciertos temores. Cada vez, las aplicaciones FLOSS se emplean más en el software de desarrollo y en los principales productos de las compañías [39]. Además, la cantidad disponible de proyectos FLOSS es enorme y, a menudo su calidad puede que no sea adecuada para la adopción en procesos de negocios reales, por tanto, las empresas deben analizar cuidadosamente las soluciones disponibles y elegir el software que se adapte a sus necesidades funcionales y de calidad [40], la cual es entendida como el "*conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia. La calidad es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, portabilidad, usabilidad, seguridad e integridad*" [41].

El desafío que se presenta es la elección de una determinada solución tecnológica software que cumpla con un requerimiento o necesidad en particular [40], en el caso del software libre [42] es aún más complejo, dados los mitos existentes en torno a él y a la dinámica en su desarrollo, que va ligada a una comunidad de desarrolladores y de usuarios, lo que hace imprescindible contar con algún método que permita hacer una evaluación cualitativa mediante la integración de las características del software libre o de código abierto. Esto ha motivado que en los últimos años se realicen múltiples investigaciones que han permitido desarrollar unas iniciativas [39] en las que se han definido elementos más precisos para la toma de decisiones y provisión de mecanismos que permitan a las comunidades de desarrollo gestionar la calidad de sus proyectos FLOSS. Esto ha ocurrido en razón a la imposibilidad que se presenta con los modelos tradicionales de calidad que se emplean en el desarrollo de software privativo (Normas ISO, modelo CMMI[4], entre otras) [43]. Al menos se han identificado 20 iniciativas diferentes para la evaluación de proyectos FLOSS [39]. Las cuales se indican en la Tabla 1.1

Tabla 1.1 Métodos, Modelos, Marcos de Referencia y/o enfoques de Evaluación de FLOSS [41].

No.	Nombre	Año	Fuente	Orig.	Método
1	Capgemini Open Source Maturity Model	2003	[44]	I	Yes
2	Evaluation Framework for Open Source Software	2004	[45]	R	No
3	A Model for Comparative Assessment of Open Source Products	2004	[46] [47]	R	Yes
4	Navica Open Source Maturity Model	2004	[48]	I	Yes
5	Woods and Guliani's OSMM	2005	[49]	I	No
6	Open Business Readiness Rating (OpenBRR)	2005	[50] [51]	R/I	Yes
7	Atos Origin Method for Qualification and Selection of Open Source Software (QSOS)	2006	[52]	I	Yes
8	Evaluation Criteria for Free/Open Source Software Products	2006	[53]	R	No
9	A Quality Model for OSS Selection	2007	[54]	R	No
10	Selection Process of Open Source Software	2007	[55]	R	Yes
11	Observatory for Innovation and Technological transfer on Open Source software (OITOS)	2007	[56],[57]	R	Yes
12	Framework for OS Critical Systems Evaluation (FOCSE)	2007	[58]	R	No
13	Balanced Scorecards for OSS	2007	[59]	R	No
14	Open Business Quality Rating (OpenBQR)	2007	[60]	R	Yes
15	Evaluating OSS through Prototyping	2007	[61]	R	Yes
16	A Comprehensive Approach for Assessing Open Source Projects	2008	[62]	R	No
17	Software Quality Observatory for Open Source Software (SQO-OSS)	2008	[63]	R	Yes
18	An operational approach for selecting open source components in a software development project	2008	[64]	R	No
19	QualiPSo trustworthiness model	2008	[65] [66]	R	No
20	OpenSource Maturity Model (OMM)	2009	[67] [68]	R	No

[4] Capability Maturity Model Integration (CMMI) es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software creado por *Software Engineering Institute (SEI)* [63].

En la Tabla 1.1, las iniciativas son identificadas en orden cronológico de su publicación, la columna "Fuente" refiere a los documentos que reportaron el método, la columna "Orig." indica si la iniciativa proviene de la (I)Industria y (R)*Research*, Investigación; la columna "Método" indica si el método está bien definido con sus actividades necesarias, tareas, entradas y salidas o es un mero conjunto de criterios de evaluación. Se observa que en la Tabla 1.1 se incluyen metodologías, que son modelos y herramientas que permiten evaluar la calidad de un software o proyecto de FLOSS; algunas con un grado de desarrollo y aplicación importante y otras aún en fase de desarrollo [41].

Otra forma de ver las metodologías relacionadas en la Tabla 1.1 es mediante la división por generaciones, como lo muestra la figura 2.1:

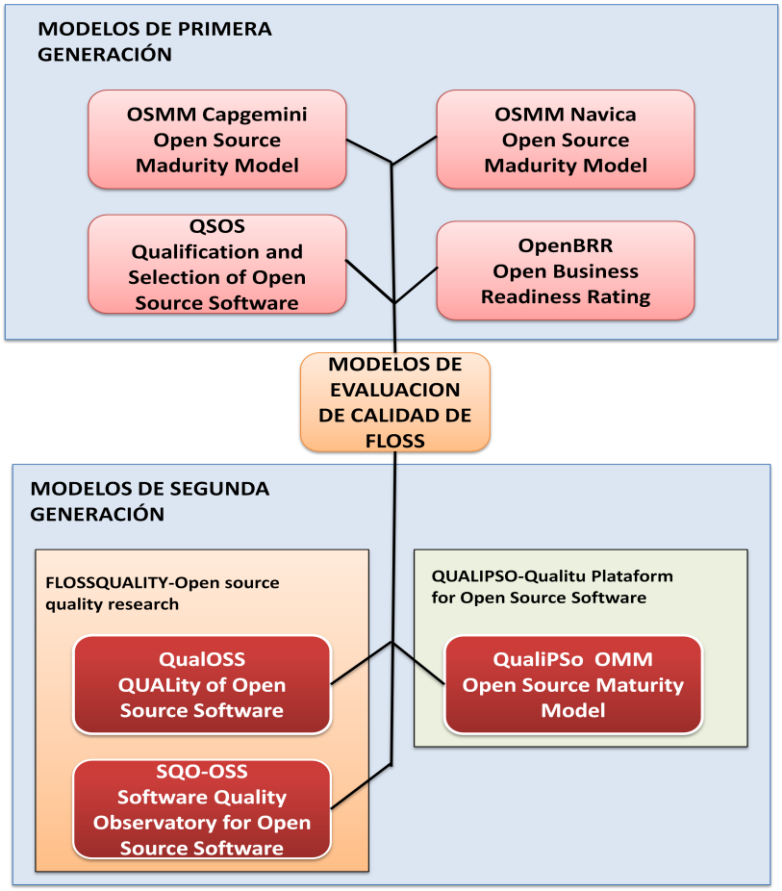

Figura 1.6 Modelos de Evaluación de Calidad de Software FLOSS

1.4.1. Modelos de Calidad de Software Libre de Primera Generación.

Entre los años 2003 y 2005 surgió la primera generación de modelos de evaluación de la calidad para software libre, estos se basaron sobre los modelos tradicionales aplicados en el software privativo con adaptaciones para el software libre, algunas metodologías que pertenecen a este conjunto son:

A. *Open Source Maturity Model*, OSMM Capgemini [34]

Este modelo es utilizado para asesoramiento independiente por parte de la empresa Capgemini, el método tiene una licencia no-libre y se requiere una distribución autorizada

[69]. Para la evaluación de un producto FLOSS, se emplean veintisiete indicadores, de los cuales, doce son los Indicadores de productos usados para determinar la madurez o adultez de un producto de FLOSS; cada indicador se califica con valores de 1, 3 y 5 [70] para determinar si un producto FLOSS es un producto inmaduro, o está en desarrollo hacia la madurez, o es suficientemente maduro [71]. Estos se agrupan en cuatro diferentes grupos:

- **El producto** (Edad, licencias, las jerarquías humanas, puntos de venta, comunidad de desarrolladores).
- **Integración** (Modularidad, colaboración con otros productos, estándar).
- **Utilización** (Soporte, la facilidad de implementación).
- **Aceptación** (Comunidad de usuarios, penetración en el mercado).

Los quince indicadores restantes, son los Indicadores de Aplicaciones empleados para determinar la adaptación de las aplicaciones FLOSS a las necesidades de los clientes [39], como son: Usabilidad, Interfaz, Rendimiento, Fiabilidad, Seguridad, Tecnología probada, Independencia de proveedor, Independencia de la plataforma, Soporte, Reporte, Administración, Asesoramiento, Capacitación, Dotación de personal e Implementación. Se les califica en una escala de 1 a 5 donde 5 es “extremadamente importante” y 1 “no importante” [70].

Todos estos datos de evaluación se combinan en una única calificación o puntuación ‘final’ que indica la idoneidad del producto para las demandas determinadas [70].

B. *Open Source Maturity Model*, OSMM de Navica

El Modelo OSMM de Navica, determina la madurez de un producto FLOSS en tres fases [40]:

Fase 1: Evaluación de la madurez del elemento de cada producto y asignación de una puntuación de madurez. Se identifican los elementos claves del producto y evalúa su madurez, estos son: Productos de Software, Soporte, Documentación, Formación, Integración de productos y Servicios profesionales. Cada elemento es evaluado y se le asigna una calificación de madurez según el siguiente proceso:

- Definición del requerimiento organizacional.
- Localización de los recursos.
- Evaluación de la madurez de elemento.
- Asignación de calificación puntuación de madurez en una escala de 1 a 10 al elemento.

Fase 2: Definición de factores de ponderación para cada elemento según los requerimientos de la organización. La tabla 1.2 proporciona una lista de calificación con valores de ponderación propuestos por defecto por el modelo, estos pueden ser cambiados de acuerdo a las necesidades específicas, solo que se debe mantener la suma de 10.

Tabla 1.2 Asignación de Valores ponderados a los factores OSSM de Navica [72]

Categoría	Ponderación
Software	4
Soporte	2
Documentación	1
Capacitación	1
Integración	1
Servicios profesionales	1
TOTAL	**10**

Fase 3: Cálculo de la calificación general de madurez del producto. OSMM asigna una ponderación a la calificación de madurez de cada elemento permitiendo que cada uno refleje su importancia a la madurez global del producto. La calificación ponderada de cada elemento se suma a proporcionar una calificación general de la madurez del producto. El modelo se suministra bajo la licencia *Academic Free License* [56].

C. Qualification and Selection of Open Source Software QSOS [52]

Proporcionada por Atos Origin para calificar y seleccionar FLOSS bajo los términos de la licencia GNU *Free Documentation License* [57] [55]. Para el desarrollo de la evaluación se cuenta con una herramienta libre llamada O3S (*Open Source Software de Selección*)[5]. Se distinguen cuatro pasos:

1. **Definición:** Identifica los factores que puedan ser considerados en los siguientes pasos de la metodología, se evalúan, entre otros, el tipo de licencia tanto actual como futura (En caso del desarrollo de producto nuevo) y los tipos de comunidades desarrolladoras.
2. **Evaluación:** Recopila información desde la comunidad FLOSS con el fin de: Construir la Tarjeta de Identidad (IC) para cada software y la hoja de evaluación del software con información general, disponibilidad de servicios, especificaciones físicas y técnicas, entre otras. Los aspectos de calidad son ponderados en un rango de 0 a 2.
3. **Calificación:** Definición de los filtros de la interpretación de las necesidades y limitaciones relacionadas con la selección de software de código libre o abierto en un contexto específico. Se logra calificar el contexto del usuario que más adelante será usado, se establece los parámetros como la necesidad del usuario de una familia o compatibilidad con un determinado sistema operativo, permitiendo filtrar lo que es realmente importante. La herramienta O3S permite la definición de estos filtros personalizados.
4. **Selección:** Identificación del software que cumple con los requisitos del usuario o comparación del software de la misma familia, esto se logra aplicando el filtro establecido en el paso C, "Calificación" de los datos, facilitados por los dos primeros pasos, con el fin de continuar las consultas, comparaciones y selección de productos.

[5] A disposición de la comunidad en el sitio Web http://www.qsos.org para crear, modificar y consultar documentos de identidad y hojas de evaluación.

D. Open Business Readiness Rating, OpenBRR

Proporcionado por *Carnegie Mellon West Center for Open Source investigation* y patrocinado por las empresas O'Reilly CodeZoo, SpikeSource, e Intel, disponible bajo la licencia "Creative Commons Attribution-NonCommercial-ShareAlike 2.5 License", es destinado a ayudar a los administradores TI a determinar qué FLOSS sería el más adecuado para sus necesidades. Se proponen estandarizaciones y agrupación en categorías de los diferentes tipos de datos de evaluación. Consta de 4 fases [18]:

Fase 1: *Quick Assessment Filter* - Evaluación Rápida

- Identificación de una lista corta inicial de los productos de software a evaluar.
- Medición de cada producto de software contra los criterios de evaluación rápida.
- Eliminación de cualquier producto de software de la lista que no cumpla con el requerimiento de usuario.

Fase 2: *Targe Usage Assessment* - Clasificación y ponderación de los criterios de selección:

Clasificación de las 12 Categorías de Evaluación de acuerdo a su importancia (1-mayor, 12-menor) y elección de 7 (o menos) de estas, para luego asignar un porcentaje de importancia a cada una, de tal forma que se totalice 100% sobre las categorías elegidas. Para cada una de las métricas dentro de una categoría, se le sitúa según la importancia a disposición de empresas. Por otra parte, ante la base de aportes de la comunidad y comentarios del público sobre el RFC, según referencia [50], se elaboró un borrador en el que refleja la retroalimentación, y en el que se reducía las categorías de 12 a 7 [51] [73].

Fase 3: *Data Colletion & Processing* - Recopilación y procesamiento de datos:

Para cada métrica utilizada en la calificación de cada categoría se calcula la ponderación aplicada para cada una. Incluye la normalización de las métricas, clasificación de categoría, factores de ponderación y mediciones de la funcionalidad de procesamiento.

Fase 4: *Data Traslation* - Cálculo y publicación de los resultados:

Clasificación de la categoría de uso de traducción de datos y la orientación funcional; y la ponderación de factores para calcular la puntuación de calificación de disposición de negocios.

1.4.2. Modelos de Calidad de Software Libre de Segunda Generación

Recientemente, ha surgido una ola de segunda generación de modelos de calidad del software libre, los cuales están basados en las metodologías tanto en los modelos tradicionales de calidad como los de primera generación del software libre. La principal diferencia es la presencia de amplia cantidad de herramientas de soporte o apoyo. Algunos modelos de calidad de segunda generación son:

A. Modelo de Calidad "QUALity of Open Source Software – QualOSS"

Es uno de los proyectos financiados por la Comisión Europea dentro del 6to programa marco [74] en relación con las métricas de software libre y de calidad, coordinado por el CETIC con participación de ocho socios de 5 países europeos: Bélgica, Francia, Alemania, España y los Países Bajos [75] para evaluar la robustez (capacidad de manejar los problemas) y la capacidad

de evolución (la capacidad para seguir siendo viable en el largo plazo) de las diversas iniciativas FLOSS para mejorar la posición competitiva de la industria europea del software, proporcionando metodologías y herramientas para mejorar la productividad y la calidad de sus productos de software. El método de evaluación QualOSS toma ventaja de la amplia información disponible en repositorios de FLOSS como los datos del software y otros datos producidos por la comunidad de desarrolladores [76].

QualOSS usa los GQM (Goal-Question-Metrics) [72] [76] y la ejecución de trazabilidad garantizan que un método de evaluación QualOSS sea riguroso. Sin embargo, un método de evaluación QualOSS debe ser documentado y las evaluaciones deben mantener un seguimiento detallado de sus distintas acciones. Así, los resultados de evaluación de QualOSS son considerados solamente válidos si se remonta a la entrada original y si todo el proceso para obtener los resultados se encuentra documentado. El proceso de Evaluación estándar se compone de cinco pasos [76]:

1. **Inicio:** identificación de la razón del negocio para la evaluación.
2. **Configuración de la evaluación:** Identificación de datos, herramientas, personas, flujos de trabajos que se involucran en una evaluación.
3. **Recolección y análisis de datos**
4. **Interpretación de los datos.**
5. **Supervisión de la evaluación.**

B. QualiPSo Open Source Maturity Model (OMM)

Modelo perteneciente a QualiPSo, uno de los proyectos financiados por la Comisión Europea dentro del 6to programa marco [74] para desarrollar una plataforma de calidad de software de código abierto centrado en la confianza y en la calidad de los sistemas de código abierto. Esta metodología está basada en el modelo CMMI y esta liberado bajo la licencia Creative Commons. El objetivo del proyecto es permitir a las empresas de software, emplear software libre en la producción y en sus principales productos a fin de aumentar el interés y el número de contribuciones a los proyectos de software libre [78] [79].

OMM contiene todos los elementos que han de ser evaluados, también tiene un conjunto de normas y directrices que describen cómo llevar a cabo los procesos de evaluación, por lo que se le denomina indistintamente modelo y metodología.

Los elementos de confianza (TWE) incluidos en OMM fueron recopilados o inspirado en dos fuentes: Los FLOSS-TWEs obtenidos de una amplia encuesta realizada a los desarrolladores, usuarios e integradores de FLOSS [67] y Áreas de proceso CMMI.

OMM está organizado en niveles de madurez, cada nivel está basado en el nivel inferior e incluye sus elementos de confianza (TWE) y son:

1. **Nivel Básico**: Que puede ser alcanzado fácilmente por la adopción de unas pocas prácticas necesarias en el proceso de desarrollo de FLOSS. Los TWEs incluidos son: PDOC, STD, QTP, LCS, ENV, MST, CM, PP1, REQM y RDMP1

2. **Nivel intermedio:** Que puede lograrse mediante el cumplimiento de todos los elementos de confianza (TWE) del nivel básico y requiere elementos de confianza (TWE) del nivel intermedio que son: RDMP2, STK, PP2, PMC, TST1, DSN1 y PPQA.
3. **El nivel Avanzado:** Es el nivel más alto que los proyectos de FLOSS pueden lograr por cumplimiento de todos los elementos de confianza (TWE) a partir de elementos de confianza de los niveles básicos y media y los requeridos del nivel avanzado, los cuales son: PI, RSKM y TST2.

C. Software Quality Observatory for Open Source Software (SQO-OSS) [80]

Proyecto financiado parcialmente por la Comisión Europea dentro del 6to programa marco [74]. Fue construido para su uso en el sistema Alitheia, una herramienta de evaluación de la calidad basada en OSGi, como un sistema de apoyo de decisión basado en mediciones, por lo que la automatización con respecto a la clasificación del software en la evaluación es una de las primeras prioridades. La plataforma de SQO-OSS tiene como objetivo combinar procesos y métricas de producto, con nuevos indicadores de calidad extraídos de base de datos que actualmente están infrautilizadas. En el proceso de construcción del modelo, se ha utilizado una versión simplificada del proceso *Goal-Question-Metric* (GQM) [63] [81] [82]. El resultado es una vista de árbol jerárquico de los atributos de calidad, cuyas hojas se analizan más en métricas medidas por el sistema y utilizadas para la evaluación de los criterios seleccionados en una simple vista y que es agregar las mediciones. Se usa el perfil basado en procesos de evaluación. El perfil basado en la agregación permite categorizar la calidad de software en cuatro categorías: *Excellent (E), Good (G), Fair (F) and Poor (P)* (o en escala ordinal E>G>F>P), para lo cual se construye perfiles con ciertos valores de medidas [83].

La implementación de SQO-OSS se basa en una arquitectura de *plugin*, que consta de componentes que se comunican a través de una bodega de datos común. Todo lo que puede ser un complemento será un *plug-in*, lo cual, significa que se tiene:

- *Plugins* para el cálculo de indicadores, procesamiento de datos y procesamiento de resultados.
- Un *plugin* para el gestor de la inserción, eliminación, y las actualizaciones de *plugins.*
- Una base de datos para hacer el seguimiento de los *plugins,* las mediciones y los datos en bruto.

SQO-OSS pretende correlacionar los datos de OSS de diversas fuentes de información con las características de calidad [80]: Funcionalidad, confiabilidad, usabilidad, eficiencia, mantenimiento y portabilidad. También tiene la intención de cooperar con otros proyectos relacionados, permitiendo que todas las partes involucradas obtengan resultados más rápidos, uniendo esfuerzos con el fin de aprovechar el crecimiento y la adopción de software libre [80].

1.4.3. Elección de la Metodología

Para la elección de la metodología de evaluación de software libre se ha efectuado una revisión de las metodologías más importantes referenciadas en las diferentes fuentes de consulta desde 2005 al 2011. Al encontrar una gran cantidad de ellas, se hace necesario contar con una forma que

permita escoger un modelo o metodología; en la revisión de consulta se encontró una propuesta, referencia [39] - *Framework for Comparing Open Source software Evaluation Methods* (FOCOSEM), según los proponentes, el objetivo de este marco no es hacer ningún juicio sobre la calidad de los métodos de evaluación de FLOSS existentes sino el de ofrecer ideas que ayuden a los profesionales a seleccionar un método adecuado de evaluación de FLOSS.

El Marco de comparación de Métodos de Evaluación de Software Libre FOCOSEM [84] está basado en cuatro fuentes diferentes: NIMSAD, que es un marco general para la compresión y evaluación de cualquier metodología, FOCSAAM, que es un marco de comparación de métodos de análisis de arquitectura de software, el marco de comparación de métodos de diseño de arquitectura de líneas de productos software y la identificación de diferencias y similitudes existentes en los diferentes métodos de evaluación de software libre [84].

Tabla 1.3 Marco de comparación de Métodos de Evaluación de Software Libre - *Framework for Comparing Open Source software Evaluation Methods (FOCOSEM)* [84].

Componente	Elemento	Descripción Breve
Contexto del Método	Objetivo específico	¿Cuál es el objetivo particular del método?
	Evaluación de la Funcionalidad	¿Es la política de funcionalidad parte del método de evaluación?
	Disponibilidad pública de resultados	¿Están las evaluaciones de los productos OSS almacenados en un repositorio de acceso público?
	Relación con otros métodos	¿Cómo se relaciona el método con otros métodos?, Es decir, ¿Qué métodos se basan en este método?
Usuario del Método	Habilidades requeridas	¿Qué habilidades necesita el usuario para emplear el método?
	Usuarios potenciales	¿Quiénes son los usuarios potenciales del método?
Proceso del Método	Actividades del Método	¿Cuáles son las actividades del método evaluación y los pasos?
	Número de criterios	¿Cuántos criterios se utilizan en la evaluación?
	Categoría de Evaluación	¿Cuáles son las categorías de criterios del método basados en las que se evalúa el producto OSS?
	Resultados	¿Cuáles son los resultados del método de evaluación?
	Herramientas de Apoyo	¿Está el método de evaluación apoyado por una herramienta?
Evaluación del Método	Validación	¿Ha sido validado el método evaluación?
	Etapa de madurez	¿Cuál es el estado de madurez del método de evaluación?

A continuación se detalla los criterios de selección que conforma al marco FOCOSEM [85] (Tabla 1.3):

A. Contexto del método: Contiene los elementos de comparación que señalan el contexto o situación en el que un método de evaluación de software libre va a ser empleado:

1. **Objetivo específico del método de evaluación:** Aunque todos los métodos de evaluación de software libre comparten el mismo objetivo general, cada método tiene un enfoque diferente por lo que es importante determinar el objetivo del método evaluado a fin de seleccionar el método adecuado según el tipo de evaluación que se realice.

2. **Evaluación de la Funcionalidad**: Algunos de los métodos de evaluación de Software FLOSS identificados se centran sólo en la calidad del producto FLOSS, mientras que otros también evalúan la funcionalidad. Este elemento permitirá considerar si es adecuado o utilizar el método de evaluación en un contexto determinado. Un método que considera la funcionalidad como criterio supone que existe un cierto nivel de flexibilidad con respecto a la funcionalidad requerida del producto a ser seleccionado, en caso de que no sea considerada la funcionalidad como un criterio de selección se puede asumir que la funcionalidad del producto está siendo evaluada conforme al requerimiento.
3. **Disponibilidad pública de resultados**: Es útil tener disponibles los resultados de una evaluación realizada previamente por otros evaluadores, con el transcurrir del tiempo, el número de resultados de la evaluación desarrollado por diferentes usuarios se va incrementando llegando a convertirse en un banco de conocimiento importante sobre determinado producto de software libre. Además, la disponibilidad de estos resultados puede permitir el ahorro de esfuerzo a un usuario que desarrolla la evaluación o si el usuario prefiere evaluar los productos de forma independiente y luego efectuar una verificación en forma cruzada. Esto puede dar al usuario una mayor confianza en los resultados obtenidos si son similares o en caso contrario ser elementos para mirar en más detalle el producto que se evalúa.
4. **Relación con otros métodos**: Comprender la relación entre los diferentes métodos es de gran importancia a fin que los evaluadores identifiquen similitudes y/o diferencias entre los métodos relacionados y seleccionar el más adecuado al requerimiento que se tiene. La relación entre los métodos de evaluación, por lo general no es explicita y por tal motivo debe ser extraída por el evaluador, se divide en 2 tipos: los “mencionados”, en los cuales, el método que se evalúa, se hace mención o referencia de otros métodos e incluso se analiza sus limitaciones; y los “basados en”, en los cuales, el método evaluado se basa en otros.

B. **Usuario del Método:** El segundo componente contiene elementos de comparación con respecto a que tipo usuarios es dirigido el método:

1. **Habilidades requeridas:** El previo conocimiento sobre las habilidades o destrezas que deba tener el evaluador al usar un método de evaluación de Software libre es útil para su escogencia, algunos métodos requerirán niveles altos de habilidad o destreza de un evaluador calificado mientras que otros requerirá de un nivel de habilidad de un usuario promedio.
2. **Usuarios potenciales:** Es importante conocer cuántos van a ser las partes o actores y el rol a desempeñar dentro del proceso definido en el método de evaluación de software libre, este elemento es importante en la escogencia del método de evaluación ya que se tiene en cuenta los recursos (humanos, financieros, etc.) disponibles para la evaluación de FLOSS.

C. **Proceso del Método:** Este componente contiene elementos para comparar la forma en que los métodos enfocan o abordan la evaluación de FLOSS.

1. **Actividades del Método:** En todos los métodos se define una secuencia de actividades, o pasos, que por lo general está organizada en un número de fases de evaluación. El número, la naturaleza y la granularidad de estos pasos pueden variar significativamente de acuerdo a un enfoque determinado. La comprensión de cuáles son las actividades involucradas pueden ayudar al profesional a evaluar la conveniencia del método en un contexto determinado.
2. **Número de criterios:** El número de criterios que se emplean para evaluar un determinado producto de software libre puede ser indicador sobre el rigor de la evaluación. No necesariamente el número de criterios puede estar relacionado con la calidad o utilidad de un método de evaluación.
3. **Categoría de Evaluación:** El número de categorías de criterios de evaluación utilizados para evaluar un producto FLOSS son indicadores sobre los muchos aspectos que se consideran que hay en el producto objeto de la evaluación. La granularidad de categorías varía de acuerdo con el método de evaluación y también depende del número de niveles de jerarquía de los criterios de evaluación.
4. **Resultados:** El resultado que se obtiene de un método de evaluación de software libre determinará la selección o no de un determinado producto FLOSS. Este resultado puede ser tan simple como un único valor o más compleja como un vector con varios valores, de todas formas, este resultado ayudará al evaluador a interpretar y tomar decisiones. Es posible que un único valor no pueda transmitir suficiente información y justificación en la que se base la toma de decisión para la escogencia de un producto de software libre.
5. **Herramientas de Apoyo:** En este elemento se tiene en cuenta la disponibilidad de un sistema de software que cuente el modelo de evaluación para apoyar a los usuarios finales en el proceso de evaluación. Algunos métodos de evaluación cuentan con estas herramientas adecuadas permitiendo acelerar el proceso de evaluación. Las herramientas pueden también permitir el almacenamiento de los resultados de la evaluación en un repositorio para su posterior reúso. Las herramientas pueden ser tan simple como una hoja de cálculo o tan complejo como un sistema de evaluación basado en web para soportar a los actores evaluadores distribuidos geográficamente. Las Plantillas de hojas de evaluación (aquellas creadas en un procesador de texto) no se consideran como herramientas.

D. Evaluación del Método: El componente de evaluación de FOCOSEM contiene dos elementos: la validación y la etapa de madurez:

1. **Validación:** La mera propuesta de un método, marco o enfoque no es suficiente, sino que debe ser validado para aumentar la confianza de los profesionales en la corrección de una solución. El investigador define tres etapas de la validación:
   - No hay validación
   - Validación limitada (por ejemplo, a través de un estudio de un caso de ejemplo que se presenta en un documento),
   - Amplia validación (método ha sido validado en varias situaciones del mundo real).

El nivel de la validación puede afectar la confianza que los usuarios de métodos pueden tener en la capacidad de ese método para satisfacer sus objetivos de evaluación.

2. **Etapa de madurez:** Se definen cuatro etapas:
    - Inicio (método ha surgido recientemente y es poco o nada utilizado);
    - Activo (método es utilizado activamente);
    - Inactivo (método todavía se admite pero no se utiliza activamente);
    - Latente (método ya no es compatible o usados).

En la medida que los investigadores han avanzado en las evaluaciones han efectuado algunos ajustes necesarios como la adición o retiro de elementos al método propuesto. Esta iniciativa ha sido publicada recientemente en una disertación de Tesis Doctoral [85] y se presentaron los resultados de comparación de seis métodos de evaluación.

Cuando esta iniciativa aún se encontraba en desarrollo y no se tenía conocimientos de publicaciones de avances o resultados obtenidos, se procedió a tomar este marco con algunas modificaciones y algunos estudios comparativos [86] [87] y se elaboró un cuadro comparativo con las metodologías: QualiPSo OMM, QualOSS, OpenBRR y QSOS que se muestra en la Tabla 1.4, a fin de clarificar algunas de las ventajas que podría presentar para el proyecto “Solución de Alta Disponibilidad (HA) y balanceo de carga para el Servicio Web de la Red de Datos de la Universidad del Cauca”.

Tabla 1.4 Comparativo de algunos modelos de evaluación empleando el marco de comparación FOCOSEM [88]

Componente	Elemento	QSOS	OpenBRR	QUALOSS	OMM
Contexto del Método	Objetivo específico	Evaluar, seleccionar y comparar FLOSS de forma objetiva, trazable con argumento sostenido.	Permitir a toda la comunidad (empresas, integradores y desarrolladores) ponderar el Software de manera abierta y estandarizada.	Permitir la comparación de productos de software libre de forma semi-automática, sencilla y rápida.	Proporcionar una base para el desarrollo de productos de manera eficiente y confiable y para evaluar los procesos empleados por las comunidades desarrolladoras e integradoras
	Evaluación de la Funcionalidad	No	Si	Si	No
	Disponibilidad pública de resultados	No, en repositorios. Si en la página web	No	Si, en repositorios.	Si, en repositorios.
	Relación con otros métodos	CapGemisis OSMM y Navica OSMM	CapGemisis OSMM y Naviva OSMM	QSOS, OpenBRR	CMMI, OpenBRR, QSOS
Usuario del Método	Habilidades requeridas	Requiere conocimiento profundo sobre cada software a utilizar y sobre cada detalle necesario para la evaluación.	Requiere conocimiento profundo sobre cada software a utilizar y sobre cada detalle necesario para la evaluación.	Requiere conocimientos básicos de instalación de software libre y manejo de la herramienta propia del método	Entendimiento básico de los procesos de desarrollo y la calidad del software. Conocimiento en manejo de herramientas libres y *open source*.
	Usuarios potenciales	Usuarios, desarrolladores e ingenieros de IT	Usuarios, desarrolladores e ingenieros de IT	Cualquier usuario alrededor del mundo	Usuarios, desarrolladores e ingenieros de IT, Compañías
Proceso del Método	Actividades del Método	Pasos: 1 - Definición y organización de lo que será evaluado 2 - Evaluación del software 3 - Calificación de su evaluación 4 - Selección	Fases: Fase 1 - Evaluación Rápida Fase 2 - Evaluación de uso objetivo Fase 3 - Recolección y procesamiento de datos Fase 4 - Traducción de datos	Pasos: 1 - Inicio de la evaluación. 2 - Creación y planeación de la evaluación 3 - Recolección y procesamiento de datos 4 - Interpretación de resultados 5 - Supervisión de la evaluación	Pasos: 1. Decisión a nivel de política corporativa/administración para incluir FLOSS. 2. Proceso de selección. 3. Designación responsable para la evaluación y selección. 4. Documentación de los requisitos/funcionalidad a ser satisfecha.
	Número de criterios	41	29	60	12
	Categoría de Evaluación	Las categorías de los criterios son: Durabilidad, Solución Industrializada, Adaptabilidad Técnica, estrategia y proveedor del servicio.	Funcionalidad, Usabilidad, Calidad, Seguridad, Rendimiento, Escalabilidad, Arquitectura, Soporte, Documentación, Adopción, Comunidad y Profesionalismo	Las características de calidad son: Durabilidad, Solución Industrializada, Adaptabilidad Técnica, estrategia y proveedor del servicio.	Tres (3) Niveles de Madurez: Básico, Intermedio, Avanzado
	Resultados	Puntaje Ponderado	Puntaje Ponderado	Puntaje Ponderado	Puntaje Ponderado
	Herramientas de Apoyo	O3S (*Open Source Selection Software*)	Plantillas de evaluación	Se pretende contar con una herramienta final que automatice el proceso de evaluación	Plantillas de evaluación online y offline. Herramienta de medición semiautomática está en la etapa de Prototipo.
Evaluación del Método	Validación	Amplia validación. El modelo está probado y validado en varios proyectos de software libre.	Amplia validación. El modelo está probado y validado en varios proyectos de software libre.	Validación limitada. El modelo está probado y validado en algunos proyectos de software libre.	Validación limitada. El modelo está probado y validado en algunos proyectos de software libre.
	Etapa de madurez	Inactivo.	Inactivo.	Inicio.	Inicio.

Al revisar el cuadro comparativo de la Tabla 1.4, se puede observar que los modelos QualiPso OMM y QualOSS, están en su etapa de madurez inicial, además de esto no cuentan con herramientas finales que permitan obtención automática de las métricas; dentro del modelo QualOSS se ha avanzado en la obtención de forma automatizada de la mayoría de las métricas relacionadas con la comunidad pero presenta el inconveniente que deben ser insertados de forma manual en la Base de Datos final; mientras que la obtención de las métricas relacionadas con la documentación se buscan de forma manual en el código fuente o en la página web del proyecto. En el caso de QualiPso OMM, la herramienta de medición semiautomática aún está en la etapa de Prototipo, solo se dispone de una hoja de cálculo electrónica para hacer una evaluación manual del mismo modo como se hace para la evaluación CMMI; a pesar de que OMM se indica que es fácil y no se requiere de habilidades, se encuentra cierto grado de complejidad en algunas métricas de algunos TWE que exige del evaluador tener un profundo conocimiento en ciertos temas para poder usar el método. Así que a pesar de QualiPso OMM y QualOSS se presentan como alternativas atractivas pero por lo reciente de los mismos no se presentan como candidatos sólidos.

OpenBRR, a pesar de su inactividad, se ha constituido en la base de nuevas metodologías de evaluación y continua siendo usado en varios proyectos desarrollados al interior de las compañías, las cuales, los consideran sensibles por lo que no están disponibles al público, es así como entre las opciones restantes se decidió que la mejor opción es **emplear la metodología OpenBRR, para efectuar la evaluación de software libre**, por lo siguiente:

- Adaptabilidad de la metodología: El hecho de permitir criterios propios para la evaluación de la funcionalidad hace posible adecuarla y personalizarla a las necesidades específicas del proyecto a implementar.
- Cantidad de parámetros de evaluación mayor a 3: Hace que la evaluación sea más precisa dado que permite una ponderación más amplia.
- Facilidad de comprensión de los criterios a evaluar: Esto hace que la evaluación de las herramientas pueda desarrollarse de forma simple.
- Uso de evaluación rápida: Que permite desechar una herramienta antes de tener que realizar un proceso más largo, lo cual resulta útil para el ahorro de tiempo.
- Facilidad de manejo: Las herramientas entregadas son sencillas de usar y ponderar, haciendo que el trabajo sea ameno y sencillo.

Una vez seleccionada la metodología de evaluación de calidad OpenBRR, se procederá a su aplicación a fin de obtener un conjunto de criterios de selección sólidos, los cuales son el fruto de la experiencia de las empresas y la comunidad involucrada en su desarrollo, por lo que es posible gozar de una base sólida para lograr la elección de una solución tecnológica de alta disponibilidad y balanceo de carga adecuada para la Red de Datos.

1.5. CRITERIOS DE EVALUACIÓN

Para la determinación de estos criterios se va a hacer uso de la metodología de evaluación de calidad de software libre OpenBRR seleccionada anteriormente y que es descrita a profundidad en el anexo A, en el que se establece la necesidad de definir dos conjuntos de criterios de evaluación[6], el primero para la elaboración de la evaluación rápida y el segundo conjunto se establece para las siguientes fases, siguiendo esto misma se tiene:

1.5.1. Definición de criterios para la Fase 1: Evaluación Rápida.

La finalidad de la evaluación rápida es el ahorro de tiempo y esfuerzo para el usuario o evaluador al eliminar proyectos que son claramente inadecuados al no cumplir con los criterios mínimos para el desarrollo viable de una solución tecnológica.

Para la definición de los criterios de esta fase, se ha seguido los pasos determinados para la evaluación rápida de la metodología, descritos en el Anexo A del presente trabajo de grado, obteniéndose lo siguiente:

1. **Uso Objetivo de la aplicación**: Se evalúa el objetivo de uso de cada uno de los diferentes proyectos encontrados a fin de determinar cuáles son de misión crítica[7], esto con el fin de ser tenidos en cuenta dentro de la solución tecnológica propuesta.
2. **Selección de un grupo de indicadores de viabilidad basado en el uso de destino**: Se selecciona los siguientes criterios de viabilidad:
   - Licencia aprobada por FSF o OSI
   - Cumplimiento de estándares
   - Adoptantes referenciales
   - Disponibilidad de un soporte u organización estable
   - Revisiones efectuadas por terceros
   - Libros
   - Seguimiento por los analistas del sector

   El indicador **Implementación del lenguaje** no se tiene en cuenta en el presente proyecto dado que es indiferente el lenguaje de programación en la que este desarrollada la solución tecnológica a escoger.
3. **Añadir más indicadores de viabilidad interna a la lista si aplica**: A la lista anterior, se adicionaron los siguientes:
   - **Obtención gratuita**: En razón a que no se cuenta con un presupuesto para la adquisición, implementación y puesta en marcha de una solución tecnológica de este tipo.

[6] En el documento RFC1 original de la metodología, se habla de métricas, donde estas se definen como una propiedad medible del FLOSS, para evitar ambigüedades y continuar con la terminología empleada desde el anteproyecto y para efectos del presente proyecto se les llamara criterios de evaluación.
[7] Un "Sistema de Misión Crítica": es un sistema cuya no disponibilidad total o parcial termina afectando negativamente las utilidades de un negocio dado [89].

- **Proyecto con un último lanzamiento inferior a tres años**: Es deseable que la solución tecnológica elegida esté vigente y evolucione a futuro, por esto se evitan los proyectos de los cuales no se tenga un lanzamiento reciente que constituye un indicador de que su comunidad es poco activa.
- **Proyecto en versión estable**: Es primordial trabajar con versiones que ya se encuentren depuradas y libres de *bugs*.

Con estos indicadores adicionales se pretende lograr la implementación y puesta en marcha de una solución tecnológica económicamente viable.

4. **Crear una política de criterios de aprobación:** Se establece como política el cumplimiento o no del criterio de viabilidad, la tabla 1.5 muestra el resumen de los criterios de evaluación de la fase 1 que han sido definidos y su correspondiente política de aprobación con los que se efectuará la evaluación rápida:

Tabla 1.5 Políticas de aprobación.

CRITERIO	POLÍTICA DE APROBACIÓN
Objetivo de uso	El objetivo de uso debe ser de misión crítica.
Licencia aprobada	La licencia debe estar aprobada por la Free Software Fundation (FSF) o por la Open Source Initiative (OSI).
Cumplimiento de estándares	Los proyectos deben cumplir estándares de la industria[8].
Adoptantes referenciales	Se obliga la existencia de adoptantes del producto.
Disponibilidad de un soporte u organización estable	Necesita contar con un soporte de una organización estable.
Revisiones efectuadas por terceros	Existencia de Revisiones efectuadas por terceros.
Libros	Se evalúa la existencia de libros en los que se haga mención de la herramienta a evaluar.
Seguimiento por los analistas del sector	Disponibilidad de informes de investigación sobre el software de los analistas de las principales empresas de investigación de mercado como Gartner e IDC
Obtención gratuita	Debe ser de obtención gratuita.
Proyecto con un lanzamiento reciente.	Verificación de un lanzamiento inferior a tres años y el no anuncio del abandono del proyecto por parte de sus desarrolladores.
Versión Estable	Debe contar con el lanzamiento de una versión estable con características útiles para el proyecto a desarrollar.
RESULTADO	Para considerarse aprobado debe tener un SI en los criterios de Misión Crítica, Licencia Aprobada, obtención gratuita, último lanzamiento inferior a tres años y versión estable; los otros criterios deben tener un 75% de respuestas afirmativas para considerarlo aprobado[9].

Nota: En los casos en los que no se dispone de la información requerida se colocará un signo de interrogación en el campo correspondiente de la tabla de evaluación de la fase 1.

[8] Cuando se habla de estándares de la industria se hace referencia a protocolos, especificaciones, RFC, IEEE, interfaces, etc. [50].
[9] La determinación de un 75% se hace siguiendo ejemplos encontrados de implementación, ya que la falta de tanta información o la negativa en la misma pueden ser un indicador de falta de documentación o importancia del proyecto.

1.5.2. Fase 2: Evaluación de Uso Objetivo

En esta fase se determina la importancia relativa de las categorías[10], a fin de determinar la jerarquía de las mismas y establecer los criterios necesarios, como se explicará más adelante, para llegar a la evaluación de calidad de la solución tecnológica planteada.

De acuerdo con el anexo A, los pasos a seguir para la elección de las categorías y posterior ponderación fueron:

1. En reunión con el Ingeniero Fabián Mera (Coordinador del área de servidores a la fecha), según su experiencia fueron ordenadas las categorías de evaluación quedando las mismas listadas de acuerdo a la importancia o relevancia (Ordenándolas de mayor a menor) que puedan tener dentro del proyecto, luego se hizo un comparativo con diferentes ejemplos de aplicación de la metodología en proyectos de misión crítica que se encontraron y/o facilitaron, se realizaron los ajustes necesarios, quedando como resultado de esta organización lo mostrado en la tabla 1.6.

Tabla 1.6 Organización de las categorías según su importancia para el proyecto.

CATEGORIA DE EVALUACIÓN	DESCRIPCION	ORDEN DE IMPORTANCIA
Funcionalidad	¿Qué tan bien se cumplen los requisitos del software del usuario?	1
Seguridad	¿Hasta qué punto el software maneja los problemas de seguridad? ¿Qué tan seguro es?	2
Comunidad	¿Cuán activa es la comunidad del software?	3
Soporte	¿Qué tan bien está soportado el software?	4
Documentación	¿Cuál es la calidad de la documentación para el software?	5
Calidad	¿De qué calidad son el diseño, el código, y las pruebas? ¿Qué tan completos y sin errores son?	6
Escalabilidad	¿En qué medida escala el software en un entorno de gran tamaño?	7
Usabilidad	¿Qué tan buena es la interfaz de usuario?, ¿Qué tan fácil es utilizar el software para los usuarios finales?, ¿Qué tan fácil es el software para instalar, configurar e implementar?	8
Arquitectura	¿Qué tan buena es la arquitectura del software? ¿Qué tan modular, portátil, flexible, abierta, extensible, y fácil de integrar es?	9
Rendimiento	¿Hasta qué punto es bueno el desempeño del software?	10
Adopción	¿Qué tan buena es la aceptación del componente por la comunidad, el mercado y la industria?	11
Profesionalismo	¿Cuál es el nivel de profesionalidad del proceso de desarrollo y de la organización del proyecto en su conjunto?	12

[10] Las categorías de evaluación se definen en la metodología para la organización del proceso de evaluación en áreas de interés y agrupación de los criterios o métricas que miden los mismos aspectos.

2. Según se indica en el Anexo A, se realiza el punto de corte escogiendo las siete (7) categorías superiores de las anteriormente listadas, quedando así: Funcionalidad, Seguridad, Comunidad, Soporte, Documentación, Calidad y Escalabilidad como categorías a ser evaluadas, dejando por fuera las cinco (5) categorías inferiores del listado que son: Usabilidad, Arquitectura, Rendimiento, Adopción y Profesionalismo; de las categorías que no van a ser evaluadas dentro de la metodología es posible observar:

- Es importante aclarar que la evaluación de la usabilidad dedica sus esfuerzos principalmente a la evaluación de la interfaz de usuario, pero dado que la solución tecnológica debe ser diseñada para funcionar en servidores y la recomendación general de los mismo es quitar el entorno gráfico para con esto reducir el consumo de recursos del sistema, además del hecho de que las medidas concernientes a la configuración depende en gran medida de las habilidades del usuario que realice dicho proceso, por lo que todo puede resultar en medidas subjetiva, y poco prácticas para el entorno de trabajo, por eso se considera que con la eliminación de esta categoría es posible una elección imparcial y centrada en la necesidad propia del proyecto.
- Arquitectura: Con esta categoría lo que se busca medir es la existencia de *plugins* desarrollados por terceros y APIS, lo cual puede resultar interesante, aun así en el caso de los *plugins* al tratarse de un desarrollo de terceros la instalación de los mismos puede constituirse en un fallo de seguridad para la solución tecnológica desarrollada.
- Rendimiento: A pesar de su importancia, cabe aclarar que el mismo es medido dentro de funcionalidad con criterios propios que permiten realizar una medida acorde a las necesidades del proyecto.
- Adopción: Su eliminación no trae consecuencias extremas ya que la misma fue implícitamente tomada mediante la revisión de documentación para la búsqueda de proyectos de Software Libre candidatos, buscándose los que han tenido mayor relevancia en el medio y por tanto siendo aquellos que cuentan con una mayor tasa de adopción.
- Profesionalismo a pesar de ser un factor importante se asume que para que una comunidad sea estable y por tanto un proyecto sea maduro debe tener criterios serios para la elección de las personas que entran a la misma y por esto es que profesionalismo puede ser descartado como categoría a evaluar.

3. Una vez realizado el proceso anteriormente descrito se procedió a elegir los valores de ponderación siguiendo el método descrito en el Anexo A, queda como resultado de este proceso la tabla 1.7.

Tabla 1.7 Valores de ponderación de las categorías OpenBRR.

CATEGORIA DE EVALUACIÓN	FACTOR DE COMPARACION
Funcionalidad	25%
Seguridad	20%
Comunidad	20%
Soporte	15%
Calidad	10%
Documentación	5%
Escalabilidad	5%

1.5.3. Establecimiento de los criterios de evaluación de las categorías

Luego de definir 7 categorías, es necesario establecer los criterios para evaluar cada una de ellas, los criterios que evaluarán la mayoría de las categorías han sido desarrollados por la comunidad y empresas encargadas de la definición de la metodología [50], excepto la categoría funcionalidad, la cual debe ser definida por el usuario de la metodología y esta debe corresponder a criterios específicos para cada proyecto; para esto se realizó un listado de criterios a evaluar, correspondientes a las funcionalidades mínimas con que debe contar la solución tecnológica, para su diseño se usaron criterios propios, lectura de documentación y acuerdos basados en la experiencia del Ingeniero Fabián Mera, se estableció también las categorías que cuentan con mayor importancia asignándoles a las mismas un 3, luego las de importancia media asignándoles un 2 y finalmente a las de importancia baja se les asigno un 1, quedando como resultado la tabla 1.8.

En la determinación de los criterios de funcionalidad, es muy importante tener en cuenta, la distinción entre el tipo de tráfico y el componente que presenta la falla, esto obedece a que el tiempo de respuesta en estas condiciones puede presentar una importante variación, se asigna mayor importancia al tráfico alto ya que es un momento muy crítico en el que un segundo puede significar mayor número de conexiones perdidas, la elección del parámetro de tiempo en 4 segundos se tomó de la tabla 3.7 de [90], en la que se relaciona los diferentes rangos de respuesta que debe tener un servicio, considerando aceptable un rango entre 2 y 4 segundos. El disminuir esos tiempos o tener otras características que no fueron tenidas en cuenta en esta lista, será “premiado”, pero la falta de cumplimiento de uno de ellos será “castigado” al restar los puntos concernientes a dicho incumplimiento.

Tabla 1.8 Criterios para medir la Categoría funcionalidad en proyectos de Alta Disponibilidad y Balanceo de Carga.

CRITERIO	PONDERACIÓN
Presenta un tiempo de caída menor a 4 segundos en caso de caída del nodo principal de balanceo de carga, durante un momento de tráfico alto.	3
Presenta un tiempo de caída menor a 4 segundos en caso de caída de un nodo en el servidor web, durante un momento de tráfico alto.	3
Presenta un tiempo de caída menor a 4 segundos en caso de caída de un nodo de la base de datos, durante un momento de tráfico alto.	3
Presenta un tiempo de caída menor a 4 segundos en caso de caída del nodo principal de balanceo de carga, durante un momento de tráfico medio.	2
Presenta un tiempo de caída menor a 4 segundos en caso de caída de un nodo en el servidor web, durante un momento de tráfico medio.	2
Presenta un tiempo de caída menor a 4 segundos en caso de caída de un nodo de la base de datos, durante un momento de tráfico medio.	2
Presenta un tiempo de caída menor a 4 segundos en caso de caída del nodo principal de balanceo de carga, durante un momento de tráfico bajo.	1
Presenta un tiempo de caída menor a 4 segundos en caso de caída de un nodo en el servidor web, durante un momento de tráfico bajo.	1
Presenta un tiempo de caída menor a 4 segundos en caso de caída de un nodo de la base de datos, durante un momento de tráfico bajo.	1
Mantiene el porcentaje de Procesamiento igual en los nodos, diferentes del balanceador.	3
Puede Manejar más de 400 conexiones simultáneas.	3
Puede contener más de 2000 conexiones sin experimentar una caída	3
Nivel de uso del procesador inferior al 10% con tráfico bajo.	2
Encripta los mensajes de comunicación	2
Puede usarse para controlar más de 10 Nodos.	1
Instalable en la distribución Linux Debian	1

Para los criterios del resto de las categorías, se determinan de acuerdo al anexo A y al Apéndice 1 del RFC1 [50], esto son:

Tabla 1.9 Criterios de medición para OpenBRR

CATEGORIA	CRITERIOS	DESCRIPCION	PONDERACION	PUNTAJE				
				5 EXCELENTE	4 BUENO	3 ACEPTABLE	2 POBRE	1 INACEPTABLE
SEGURIDAD	El número de vulnerabilidades de seguridad en los últimos 6 meses que son desde moderadas a extremadamente críticas	Este mide la calidad relacionada con las vulnerabilidades de seguridad. ¿Cómo es susceptible el software a vulnerabilidades de seguridad?	35%	0	1 – 2	3 – 4	5 – 6	> 6
	El número de vulnerabilidades de seguridad que siguen abiertas (sin parchear)	Esto mide, si el proyecto es capaz de resolver todos los problemas de seguridad.	35%	0	1	2	3 – 5	> 5
	¿Hay una página dedicada a la información (página web, wiki, etc) para la seguridad?	Este mide la conciencia y seriedad con la que el proyecto toma los problemas de seguridad.	30%	Si, en buen estado		Si		No
COMUNIDAD	El volumen medio de la lista de correo general en los últimos 6 meses	La lista de correo general es el lugar donde la comunidad se ayuda.	50%	> 720 mensajes per mes	300 - 720 msg por mes	150 - 300 msg por mes	30 - 150 msg por mes	< 30 msg por mes
	Número de contribuyentes de código en los últimos 6 meses	Los contribuyentes de código por lo general promueven la construcción de la comunidad en torno al proyecto. Cuanto mayor sea el número de contribuyentes de código, mejor será el apoyo de la comunidad.	50%	> 50	20 – 50	10 - 20	5 - 10	< 5
SOPORTE	El volumen medio de la lista de correo general en los últimos 6 meses	La lista de correo general, es el primer lugar donde la gente va en busca de ayuda gratuita.	50%	> 720 mensajes por mes	300 – 720 msg por mes	150 – 300 msg por mes	30 – 150 msg por mes	< 30 msg por mes
	La calidad del apoyo profesional	El apoyo profesional que ayude a afinar para el despliegue local y la solución de problemas es siempre deseable.	50%	Instalación + solución de problemas + Integración / Apoyo personalizado		Solo Soporte de Instalación.		No soporte profesional.
DOCUMENTACIÓN	La existencia de varios tipos de documentación	Una suite de una buena documentación debe incluir la documentación de varios grupos de usuarios en varios formatos.	50%	Instalar / implementar, usuario, administración, optimización, mejora, desarrollo, la documentación está disponible en múltiples formatos (pdf, html sola, de	Instalar / despliegue, usuario, administrador, guías de actualización disponibles en varios	Instalar / implementar y guía de usuario están disponible	Sólo basado en texto, existe documentación de instalación	Sin la documentación adecuada. Un archivo README no cuenta

CATEGORIA	CRITERIOS	DESCRIPCION	PONDERACION	PUNTAJE				
				5 EXCELENTE	4 BUENO	3 ACEPTABLE	2 POBRE	1 INACEPTABLE
				varios archivos html).	formatos			
	Marco de Contribución de Usuarios.	Los mejores guías a menudo provienen de la entrada /muestras del usuario. Esta es la opinión de personas que han utilizado los productos.	50%	Las personas pueden contribuir, y las contribuciones se editan/filtran por los expertos		A las personas se les permite contribuir		Los usuarios no pueden contribuir.
CALIDAD	Número de versiones menores en los últimos 12 meses	Esto mide las actualizaciones planificadas y correcciones de errores. Normalmente, paquetes de servicio en productos comerciales.	5%	2		1 o 3		0 o > 3
	Número de comunicado de point/patch en los últimos 12meses	Normalmente, los oficiales de point/patch liberados son correcciones de errores P1 como bloqueos, memoria y vulnerabilidades de seguridad.	20%	3 – 4		1 – 2, o 5 – 6		0 o > 6
	Número de bugs abiertos en los últimos 6 meses	Este mide la calidad de uso del producto.	15%	< 50	50 – 100	100 – 500	500 – 1000	> 1000
	Número de errores corregidos en los últimos 6 meses (en comparación con el # de errores abiertos)	Este mide la rapidez con errores fijos.	20%	> 75%	60% - 75%	45% - 60%	25% - 45%	< 25%
	Número de errores P1/críticos abiertos	Este mide la gravedad de los problemas de calidad encontrados	20%	0	1 – 5	5 – 10	10 – 20	> 20
	La edad promedio de los errores de P1 en los últimos 6 meses	La edad promedio de errores de P1 en los últimos 6 meses.	20%	< 1 semana	1 – 2 semanas	2 – 3 semanas	3 – 4 semanas	> 4 semanas

CATEGORIA	CRITERIOS	DESCRIPCION	PONDERACION	PUNTAJE				
				5 EXCELENTE	4 BUENO	3 ACEPTABLE	2 POBRE	1 INACEPTABLE
ESCALABILIDAD	Referencia de implementación	Mide si el software es escalable y probado en el uso real a través de una implementación real.	50%	Si, con publicaciones de los usuarios.		Si		No
	Diseñado para la escalabilidad	Evalúa si el componente ha sido diseñado con la escalabilidad en mente. ¿Es segura para los subprocesos? ¿Se ejecuta en un entorno de clúster? Puede H / W resolver problemas de rendimiento?	50%	Si, extensivo.		Si, algunos.		No

Finalmente, al realizar los pasos anteriormente indicados, se ha obtenido el conjunto de criterios que serán usados para la validación y elección del proyecto que conformará la solución tecnológica, dándose con esto, cumplimiento al primer objetivo específico determinado en el anteproyecto de grado, el cual fue: "Definir un conjunto de criterios que permitan seleccionar y evaluar herramientas libres de Alta disponibilidad (HA) y Balanceo de Carga para el servicio Web de la Red de Datos de la Universidad del Cauca".

Durante el desarrollo de este capítulo, se ha explorado los conceptos de alta disponibilidad y balanceo de carga, dado que es un tema extenso imposibilita tratar en su totalidad y profundidad, estos conceptos brindan claridad sobre el significado, la funcionalidad, ámbito de aplicación, los requisitos técnicos y resultados esperados con su aplicación. También se ha explorado la conceptualización sobre el software Libre y/o open Source, con énfasis en la existencia de gran cantidad de métodos y metodologías de evaluación de software libre y/o Open Source con el que se permite elegir soluciones o iniciativas FLOSS. Se revisaron aquellos que con más frecuencia aparecían reseñados y mencionados en las distintas fuentes de consulta; ante tal cantidad de métodos y metodologías de evaluación, se logra encontrar y emplear un marco de comparación de metodología de evaluación de software libre denominado **FOCOSEM** con el que se selecciona el metodolo **OpenBRR**, que al usarse se obtiene un conjunto de criterios con los que se va a evaluar y seleccionar el o los proyectos FLOSS que conformará la solución tecnológica.

CAPÍTULO 2: ANÁLISIS DE LA ARQUITECTURA ACTUAL Y PLANTEAMIENTO DEL PROBLEMA

Los servicios web ofrecen la posibilidad de publicitar en una herramienta de uso global y de amplia difusión como lo es el Internet; es por esto que las aplicaciones web desempeñan un papel importante en el crecimiento y el funcionamiento de cualquier organismo, pero, no solo basta con que se encuentren adecuadamente diseñados, la disponibilidad de las mismas es tan bien, un factor importante ya que afecta la percepción del usuario final, además, el cese de actividades al interior de una organización, puede llevar a pérdidas de tipo financiero, a fin de evitar esto, es necesario efectuar una evaluación completa para encontrar los posibles puntos de falla a fin de poder hacer los correctivos adecuados ante eventualidades de fallo.

En este capítulo se explorará y evaluará el esquema sobre en la cual se encuentran desplegados el servicios Web y el Sistema Integrado de Matriculas y Control Académico (SIMCA), a fin de determinar los requerimientos necesarios que deban ser satisfechos por la solución tecnológica y con esto mejorar la disponibilidad y optimización en el uso de los recursos tanto tecnológicos como humanos.

2.1. DEFINICIONES GENERALES

2.1.1. Aplicación Web

Las aplicaciones web son soluciones informáticas o como su nombre lo dice “aplicaciones” que permiten interactuar con la información y a las cuales es posible acceder a través de una conexión a Internet o de una intranet mediante un navegador, su uso más extendido se da en la publicación de contenidos de tipo dinámico que permitan la interacción con el usuario, aunque existen muchas variaciones, una aplicación web está estructurada por lo general como una aplicación de cuatro capas (Figura2.1)

Figura 2.1 Esquema General para una aplicación Web.

Las cuatro capas ilustradas en Figura 2.2 son:

- **Balanceo de cargas:** Esta capa contiene los elementos necesarios para recibir las peticiones de los usuarios y direccionarlos hacia el servidor Web.

- **Capa de Acceso**: En esta capa se tienen configurados todos los servidores Web, estos reciben las peticiones y las reenvían hacia los servidores de aplicaciones donde se tiene implementada la lógica del negocio.
- **Capa de Aplicaciones**: Es el lugar en que se tiene desplegada la aplicación sobre los servidores, por eso es allí donde se implementa la lógica de negocio teniendo privilegios de acceso a los datos a través del motor de base de datos.
- **Persistencia:** Es en esta capa es donde los datos de la aplicación están almacenados, su función es ofrecer servicios de persistencia y recuperación de información a las capas superiores, y es sobre esta capa donde se tienen todos los datos pertenecientes a la base de datos. Aunque están aquí los servidores con los motores de Base de Datos, está enfocado más a la información de la base de datos como tal (Datafiles, Logs, entre otros) elementos que deben estar al alcance de todos los servidores de aplicaciones desplegados sobre el esquema.

2.1.2. Sitio Web con Gestor de Contenidos

Los gestores de contenido web son una categoría de aplicación web pensada únicamente para la publicación de contenidos estáticos.

Para un sitio web con gestor de contenidos, se tiene la Figura 2.2.

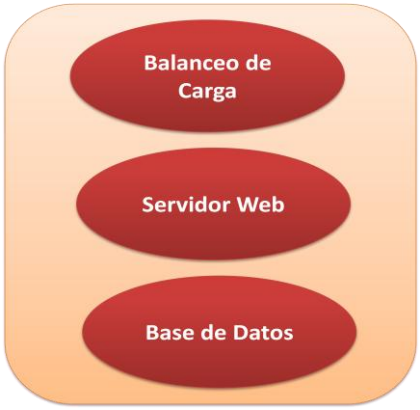

Figura 2.2 Esquema General para un sitio web con gestor de contenidos

De la figura 2.2 es posible observar 3 capas:

- **Balanceo de cargas:** Se hace necesario en sitios que experimenten alto tráfico y por ende posean más de un servidor web, en esta capa residen los elementos necesarios para recibir las peticiones de los usuarios y direccionarlos hacia los servidores Web.
- **Servidor web:** Esta capa está formada por uno o varios servidores web (también llamados servidor HTTP) siendo estos un software que procesa una aplicación del lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se utiliza el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación

del modelo OSI. El término también se emplea para referirse al computador que ejecuta el programa.

- **Base de Datos:** Esta capa contiene almacenada toda la información que se publica en el sitio y es accedida por el servidor web.

Con lo anterior en mente se va a proceder a explicar el esquema actual implementado y los diferentes problemas hallados en él.

2.2. ANÁLISIS DE LA ARQUITECTURA ACTUAL PARA SIMCA

El Sistema de Registro y Control Académico (SIMCA) puede catalogarse como una clásica aplicación web cuya funcionalidad es la de aceptar ordenes o respuestas del usuario y basado en eso hacer la búsqueda dentro de una base de datos existente.

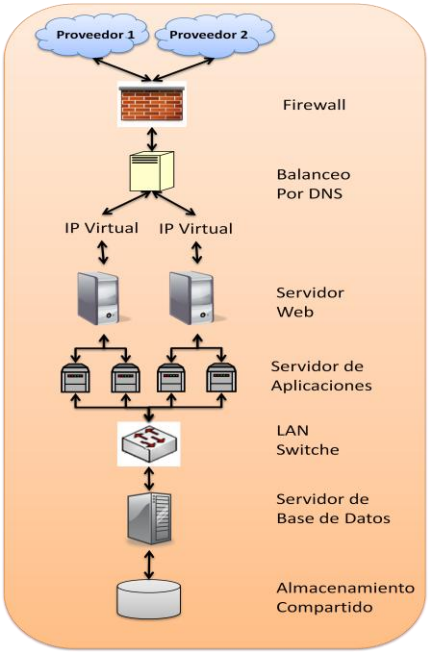

Figura 2.3 Arquitectura actual para SIMCA.

El esquema que actualmente está siendo implementado se aprecia en la Figura 2.3. , en la que se observa lo siguiente:

- **Balanceo de carga**: El balanceo de carga está siendo llevado a cabo por el servidor DNS, este método de balanceo de carga presenta dos desventajas principales:
  - ✓ **No ofrece ningún soporte verdadero para la afinidad del servidor.** La afinidad del servidor es una capacidad de balanceo de carga del sistema para manejar las peticiones de un usuario, a un servidor específico o a cualquier servidor, dependiendo de si la información de la sesión está mantenida en el servidor o en un subyacente nivel de la base de datos.

    Sin afinidad del servidor, el DNS round robin confía en uno de tres métodos ideados para mantener control de la sesión o identidad del usuario a las peticiones que vienen sobre el

HTTP, que es un protocolo sin estado, estos son: Cookies, campos ocultos o URL re-escribible.

Cuando un usuario hace una primera petición, el servidor Web devuelve una señal de texto única identificativa a ese usuario. Las peticiones posteriores incluyen esta señal usando cookies, URL re-escribible, o campos ocultos, permitiendo al servidor mantener una sesión entre el cliente y el servidor. Cuando un usuario establece una sesión con un servidor, todas las peticiones subsecuentes van generalmente al mismo servidor.

El problema es que el navegador cachea la dirección IP del servidor. Una vez que la caché expire, el navegador hace otra petición al servidor del DNS para la dirección IP asociada al Nombre de Dominio. Si el servidor DNS devuelve una dirección IP diferente, que otro servidor en el *cluster*, se pierde la información de la sesión.

- ✓ **Ningún soporte para la alta disponibilidad.** El problema radica es su incapacidad para detectar la caída de un nodo y por ende seguir enviando peticiones aun cuando esta no serán respondidas. Un enrutador avanzado soluciona este problema comprobando los nodos en unos intervalos regulares, detectando los nodos caídos y quitándolos de la lista, así que ninguna petición va a ellos. Sin embargo, el problema todavía existe si el nodo está activo pero la aplicación web que corre en él se cae.

  Aunque éste método intenta balancear el número de usuarios en cada servidor, no balancea necesariamente la carga del servidor puesto que algunos usuarios podrían efectuar más peticiones demandando una mayor carga de actividad durante su sesión que otros usuarios en otro servidor.

- **Capa de Acceso**: La arquitectura muestra dos servidores Web donde se tiene desplegado el frente de la aplicación. Sobre esta puede tenerse implementada la cantidad de servidores que sean necesarias para atender a los usuarios. Básicamente la petición que llega por parte del usuario y es direccionada a este servidor por el nodo de la capa del balanceador de cargas es recibida por medio del protocolo HTTP o HTTPS (generalmente por los puertos 80, 443 o los definidos para tal fin). Una vez se establece el canal de comunicación entre el servidor Web y el usuario se tendrá flujo de datos en ambas direcciones. Los datos recibidos por el servidor web son enviados hacia alguno de los servidores de aplicaciones que están en su dominio, donde se hace el procesamiento de esta y se genera la respuesta a la petición del cliente por medio del servidor Web.

- **Capa de Aplicación**: En esta capa se encuentra el servidor de aplicaciones Jboss y las clases java que interactúan directamente con la base de datos, para este, existe una solución tecnológica llamada Jboss *cluster*. En un clúster de JBoss Application Server (AS), (también conocido como una "partición"), un nodo es una instancia del servidor de aplicaciones JBoss. La comunicación entre los nodos está a cargo de la biblioteca de la comunicación *JGroups*, con un canal *JGroups* que proporciona la funcionalidad central de seguimiento de estado del grupo de forma fiable y el intercambio de mensajes entre los miembros del clúster. Los canales JGroups con la misma configuración y el nombre tienen la capacidad de descubrirse de forma

dinámica entre sí y formar un grupo. Esta es la razón, de que simplemente con ejecutar "run -c all" en dos Servidores de Aplicación de la misma red, es suficiente para que formen un grupo, cada AS inicia un canal (en realidad, varias) con la misma configuración por defecto, por lo que se descubren dinámicamente entre sí y forman un grupo. Los nodos se pueden agregar dinámicamente o pueden ser suprimidos de los grupos en cualquier momento, simplemente con iniciar o detener un canal con una configuración y un nombre que coincide con los otros miembros del clúster [10]. En resumen, un conjunto de JBoss es un conjunto de instancias del servidor de AS en el que en cada uno de los nodos se está ejecutando un canal configurado de forma idéntica y nombrado JGroups.

- **Capa de Persistencia**: En esta capa, se cuenta con la base de datos Oracle, que tiene a su vez un sólo servidor de base de datos, siendo por tanto un único punto de fallo, *SPOF (Single Point Of Failure),* y convirtiéndose así en un riesgo potencial para el funcionamiento conjunto del sistema.

Como conclusión de la anterior descripción y según la investigación realizada mediante la lectura de diferentes foros y averiguaciones con expertos, se evidenció que la mejor solución tecnológica para la capa de aplicación es la que está actualmente implementada, sin embargo se identificaron dos puntos potenciales de fallo, el primero de ellos se encuentra en la capa de balanceo de carga, y el segundo es hallado en la capa de persistencia, por tanto sobre estos dos puntos será diseñada la solución tecnológica, teniendo como **requerimientos iniciales** hacer un balanceo de carga para servidores de aplicación JBOSS, este balanceador debe además estar en la capacidad de detectar su propio estado, migrando el servicio a otro nodo en caso de presentarse una falla, y además deben ser implementadas también herramientas de alta disponibilidad y balanceo de carga para la base de datos Oracle.

2.3. ARQUITECTURA ACTUAL PARA EL PORTAL WEB

El portal Web de la Universidad del Cauca funciona como un gestor de contenido web, ya que está siendo utilizado principalmente para la publicación de tipo estático, el esquema que actualmente está siendo implementado se aprecia en la Figura 2.4.

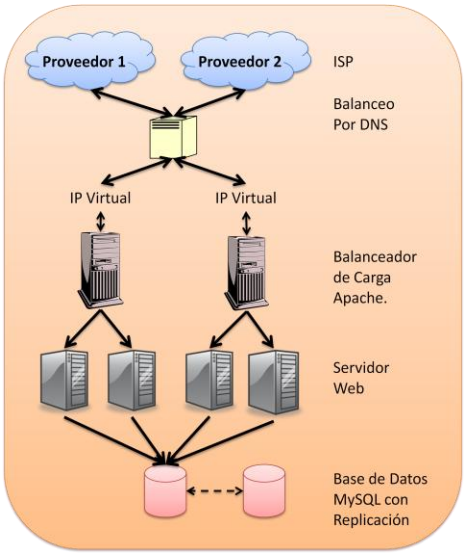

Figura 2.4 Arquitectura actual para el Portal Web.

De la figura 2.4 se observa lo siguiente:

- **Balanceo de Carga**: Al igual que para el caso anterior, se observa un balanceo de carga por DNS, y para este caso aplican las desventajas anteriormente mencionadas, adicional a esto aquí se presenta un balanceo de carga realizado con Apache y el módulo mod_jk, el cual, no cuentan con un respaldo que permita soportar un fallo, haciendo que la falta de uno de estos comprometa el rendimiento del *cluster* ya que disminuye su capacidad operativa a la mitad.
- **Servidor Web**: Implementados sobre Apache, su función es proveer la información a las computadoras que se conectan a él y de esta forma acceder a la información y los recursos que contiene. Sobre esta capa puede tenerse implementada la cantidad de servidores que sean necesarias para atender a los usuarios.
- **Base de Datos:** Esta funciona bajo MySQL, y si bien la base de datos hace uso de la replicación, es importante aclarar que la misma está programada para ser llevada a cabo en horas especificas del día, haciendo que en caso de una eventualidad se pierdan los datos ingresados en el periodo de tiempo que se dejo sin replicar, además, en caso de presentarse un incidente la inserción de la base de datos secundaria o de *backup* es llevada a cabo de forma manual, haciendo que los tiempos de caída en estos casos sean altos, esto último obedece al hecho de que el sistema de monitoreo puede demorar hasta 5 minutos en realizar el informe de la contingencia, esto sumado también con el tiempo que le tome al administrador ejecutar la inserción de la nueva base de datos.

Como conclusión, después de la anterior descripción, se denota que los problemas principales se encuentran en la base de datos y en el balanceo de carga, por esto es en esos dos puntos donde se buscará implementar una solución tecnológica, dejando lo anterior como **requerimientos iniciales** hacer un balanceo de carga para los servidores Apache, estos balanceadores además de detectar el estado de dichos servidores deberán ser capaces de detectar su propio estado migrando el

servicio a otro balanceador en caso de un evento de falla, además de ello se implementarán técnicas de alta disponibilidad para la Base de Datos MySQL.

2.4. DISPONIBILIDAD DE LOS SERVICIOS

Para determinar los estados de disponibilidad de los servicios actualmente implementados se recurrió a las estadísticas arrojadas por Nagios, un sistema de monitoreo de red implementado por la red de datos de la universidad del Cauca, el periodo de observación tomado fue de un año, arrojando así los siguientes datos:

- **Disponibilidad actual para el Sistema Integrado de Registro y Control Académico SIMCA (Figura 2.5)**:

State	Type / Reason	Time	% Total Time	% Known Time
UP	Unscheduled	301d 22h 33m 0s	82.723%	99.650%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**301d 22h 33m 0s**	**82.723%**	**99.650%**
DOWN	Unscheduled	1d 1h 27m 0s	0.291%	0.350%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**1d 1h 27m 0s**	**0.291%**	**0.350%**
UNREACHABLE	Unscheduled	0d 0h 0m 0s	0.000%	0.000%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**0d 0h 0m 0s**	**0.000%**	**0.000%**
Undetermined	Nagios Not Running	0d 0h 0m 0s	0.000%	
	Insufficient Data	62d 0h 0m 0s	16.986%	
	Total	62d 0h 0m 0s	16.986%	
All	Total	365d 0h 0m 0s	100.000%	100.000%

Figura 2.5 Porcentaje de disponibilidad para SIMCA

En la figura 2.5, se observa que el índice de disponibilidad es del 99.650%

- **Disponibilidad actual para el Portal Web (Figura 2.6):**

State	Type / Reason	Time	% Total Time	% Known Time
UP	Unscheduled	151d 19h 37m 22s	41.594%	99.880%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**151d 19h 37m 22s**	**41.594%**	**99.880%**
DOWN	Unscheduled	0d 4h 22m 38s	0.050%	0.120%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**0d 4h 22m 38s**	**0.050%**	**0.120%**
UNREACHABLE	Unscheduled	0d 0h 0m 0s	0.000%	0.000%
	Scheduled	0d 0h 0m 0s	0.000%	0.000%
	Total	**0d 0h 0m 0s**	**0.000%**	**0.000%**
Undetermined	Nagios Not Running	0d 0h 0m 0s	0.000%	
	Insufficient Data	213d 0h 0m 0s	58.356%	
	Total	213d 0h 0m 0s	58.356%	
All	Total	365d 0h 0m 0s	100.000%	100.000%

Figura 2.6 Porcentaje de disponibilidad para el Portal Web

En la figura 2.6, se observar que el índice de disponibilidad es del 99.880%

Además de los valores anteriores, mediante la revisión de los *logs* de la herramienta de monitoreo Nagios y luego de descartar aquellos tiempos de caída que correspondían a mantenimientos programados, fue posible estimar el tiempo medio entre fallas y el tiempo medio de reparación dependiendo del elemento involucrado, para el portal web se relaciona en la tabla 2.1 y para SIMCA en la tabla 2.2.

Tabla 2.1 Tiempos de Respuesta Estimados ante un fallo para el portal Web

Elemento de Fallo	Tiempo Medio Entre Fallas (En Días)	Tiempo Medio de Reparación (En Minutos)
Nodo Balanceador	91,194	2,69
Servidor de Base de Datos	85,422	6,75

Tabla 2.2 Tiempos de Respuesta Estimados ante un fallo para SIMCA

Elemento de Fallo	Tiempo Medio Entre Fallas (En Días)	Tiempo Medio de Reparación (En Minutos)
Nodo Balanceador	41,150	5,71
Servidor de Base de Datos	24,895	11,62

De los anteriores datos, es importante especificar que los valores de disponibilidad del servicio especificados por Nagios en las figuras 2.5 y 2.6 son globales e independientes del módulo que falló, aun así, es necesario comprender que algunos elementos del sistema pueden presentar fallas simultáneas, por lo cual, no necesariamente los valores de indisponibilidad de un elemento o la suma de tiempos de caída pueden ser directamente relacionados con el valor específico arrojado; se concluye que a pesar de que los índices de disponibilidad actuales son muy altos pueden ser mejorados con la implementación de herramientas que aporten mecanismos de respuesta automática ante eventualidades de fallo de los componentes que constituyen el sistema, disminuyendo los tiempos de indisponibilidad y respuestas para cada uno de ellos, aumentando así la disponibilidad del conjunto.

Durante la exploración de esta capítulo, se ha evaluado los diferentes aspectos para Alta Disponibilidad, como la existencia de puntos de fallos, falta de redundancia, etc. en las arquitecturas actuales en la que están desplegados los servicios de portal Web y el Sistema de Registro y Control Académico (SIMCA) y el resultado obtenido es que ambas arquitecturas presentan puntos de fallo en la parte de balanceo de carga y en la base de datos, por tanto no se encuentran correctamente preparadas para la mitigación de los efectos que una falla puede acarrear.

CAPÍTULO 3: DISEÑO Y EVALUACIÓN DE LA SOLUCIÓN

Es importante aclarar que la alta disponibilidad, ya sea de servicios o de datos, puede ser alcanzada haciendo uso tanto de soluciones *software* como de soluciones *hardware*. La variedad que existe en ambos grupos es muy grande, pudiendo elegir siempre una solución tecnológica que encaje perfectamente según las necesidades específicas que se tengan. Las soluciones *software* permiten alcanzar alta disponibilidad con la instalación y configuración de determinadas herramientas y aplicaciones que han sido diseñadas para tal efecto, las soluciones depende en gran medida del tipo de datos o servicios a los que se quiere dotar de esta característica, y normalmente suelen usarse para apoyar las soluciones de tipo *software* que se están aplicando, para mantener o superar el nivel de seguridad y disponibilidad de las mismas. En la mayoría de los casos se entiende como solución *hardware* para alta disponibilidad la replicación de recursos *hardware*, como memoria, almacenamiento, o la puesta en marcha de dispositivos con *hardware* especializado. Por ejemplo, "*en cuanto a alta disponibilidad de datos se refiere, soluciones como NAS, RAID, SAN, entre otros son ampliamente conocidas y usadas"* [91], pero dichas soluciones en muchos casos pueden resultar demasiado costosas haciendo que la relación costo-beneficio las haga descartables o que resulten inalcanzables para pequeñas y medianas empresas que no cuentan con un buen presupuesto, razón por la cual en el presente proyecto se decidió por la búsqueda de una solución software, particularmente del tipo libre.

El método más rentable para aumentar la confiabilidad general de un sitio es implementar un clúster de conmutación por error o clúster de Fail-over. Un *Cluster* de *fail-over* se compone de varios equipos en común, cada uno de los cuales es un candidato a servidor para los sistemas de archivos, bases de datos o aplicaciones. Cada uno de estos sistemas mediante el uso de herramientas de alta disponibilidad y/o balanceo de carga monitorea el estado de los otros sistemas en el clúster, en caso de presentarse un fallo en uno de los miembros del grupo, los otros toman los servicios del nodo que ha fallado. La toma de posesión se realiza de tal manera que sea transparente para los sistemas cliente que están accediendo a los datos [92].

En este capítulo se presentarán aquellos proyectos o software encontrado que pueden llegar a hacer parte de la solución tecnológica a implementar, además de la evaluación de los mismos basados en la metodología y criterios definidos en el Capítulo 1.

3.1. PROYECTOS LIBRES DE ALTA DISPONIBILIDAD Y BALANCEO DE CARGA

Dentro del software libre existe una variedad de herramientas que pueden llegar a ser útiles para la creación de un *cluster* de *fail-over*, pero en su mayoría, es necesario el uso de más de uno de los proyectos existentes para lograr una solución tecnológica completa, estos proyectos se pueden dividir en Proyectos de propósito general y Proyectos de propósito específico.

Según lo determinado en el capítulo 2 y luego de una revisión inicial fue posible llegar como resultado a la tabla 3.1 que lista los diferentes proyectos que pueden cumplir estas funciones, una explicación ampliada de sus características y división es posible encontrarla en el Anexo B.

Tabla 3.1 Proyectos enfocados Libres de Alta Disponibilidad y Balanceo de Carga

<table>
<tr><th>Tipo de Proyecto</th><th colspan="3">Nombre del Proyecto</th></tr>
<tr><td rowspan="19">DE PROPÓSITO GENERAL</td><td colspan="3">Keepalived</td></tr>
<tr><td rowspan="5">Linux-HA</td><td rowspan="5">Sub-proyectos de Linux HA:</td><td>Pacemaker</td></tr>
<tr><td>Heartbeat</td></tr>
<tr><td>OpenAIS</td></tr>
<tr><td>Resource Agent</td></tr>
<tr><td>Cluster Glue</td></tr>
<tr><td colspan="3">Linux Virtual Server</td></tr>
<tr><td colspan="3">Corosync Cluster Engine</td></tr>
<tr><td colspan="3">Ultramonkey</td></tr>
<tr><td colspan="3">Kimberlite</td></tr>
<tr><td colspan="3">LifeKeeper</td></tr>
<tr><td colspan="3">HP Serviceguard</td></tr>
<tr><td colspan="3">Open HA Solaris Cluster</td></tr>
<tr><td colspan="3">Veritas Cluster Server</td></tr>
<tr><td colspan="3">Red Hat Cluster Suite</td></tr>
<tr><td colspan="3">Wackamole</td></tr>
<tr><td colspan="3">Piranha</td></tr>
<tr><td colspan="3">HAProxy</td></tr>
<tr><td colspan="3">PEN Load Balancer</td></tr>
<tr><td colspan="3">Zen load balancer</td></tr>
<tr><td colspan="3">Vyatta</td></tr>
<tr><td colspan="3">POUND</td></tr>
<tr><td rowspan="2">DE PROPÓSITO ESPECÍFICO</td><td colspan="3">MySQL Cluster</td></tr>
<tr><td colspan="3">Oracle Real Application Clusters (Oracle RAC)</td></tr>
<tr><td>ENFOCADOS EN MÁQUINAS VIRTUALES</td><td colspan="3">Proxmox</td></tr>
</table>

A pesar que Oracle Real Application Clusters (Oracle RAC) no es tecnología de uso libre, se ha agregado a la lista anterior, dada la necesidad de solucionar los problemas de disponibilidad existentes para la actual base de datos Oracle sobre la cual funciona el servicio de Sistema y Registro y Control Académico (SIMCA), con los desarrolladores se habló sobre la posibilidad de migración a otras bases de datos pero se explicó que se está llevando a cabo un proceso de unificación para poner el sistema completo (adicionando finanzas y otros) sobre Oracle, razón por lo cual trabajar sobre Oracle ya es una decisión tomada y sería muy traumático pensar en una migración.

En la anterior lista, aparecen algunos proyectos que son de pago, otros son completamente gratuitos, mientras que otros ofrecen soporte o disponen de una gran comunidad, etc. En *GNU/Linux* se dispone de diversas soluciones pudiendo escoger la que más se adapte a las

necesidades específicas del proyecto, en tal caso se optará por proyectos gratuitos dadas las limitaciones económicas del mismo.

Después de generar la lista de proyectos encontrados en diversos sitios y/o repositorios, es comprensible la necesidad de emplear la metodología de evaluación de calidad de Software Libre elegida, ello con el fin de lograr una evaluación imparcial de las mismas.

3.2. EVALUACIÓN RÁPIDA.

Esta corresponde a la Fase I de la metodología OpenBRR, la finalidad de la evaluación rápida es el ahorro de tiempo y esfuerzo al eliminar proyectos que son claramente inadecuados al no cumplir con los criterios mínimos para el desarrollo viable de una solución tecnológica.

La evaluación se hará basada en la tabla 1.5 del presente proyecto en donde se definió el conjunto de criterios para esta fase de evaluación así como su respectiva política de cumplimiento, como resultado de este proceso se obtuvo la tabla 3.2 donde se listan los diferentes proyectos así como su evaluación y resultados.

Tabla 3.2 Resultado de la Evaluación Rápida

NOMBRE	OBJETIVO DE LA APLICACIÓN	USO DE DESTINO										RESULTADO
	OBJETIVO DE USO MISIÓN CRITICA	LICENCIA APROBADA	CUMPLIMIENTO DE ESTÁNDARES	ADOPTANTES REFERENCIALES	SOPORTE U ORGANIZACIÓN ESTABLE	REVISIONES EFECTUADAS POR TERCEROS	LIBROS	SEGUIMIENTO POR LOS ANALISTAS DEL SECTOR	OBTENCIÓN GRATUITA	ULTIMO LANZAMIENTO RECIENTE	VERSIÓN ESTABLE	
Wackamole	Si	No	?	Si	?	No	Si	?	Si	No	Si	No candidato
Cluster Glue	Si	Si	Si	Si	Si	Si	Si	?	Si	Si	Si	Si Candidato
Heartbeat	Si	Si	Si	Si	Si	Si	Si	?	Si	Si	Si	Si Candidato
Resourse Agent	Si	Si	Si	Si	Si	Si	Si	?	Si	Si	Si	Si Candidato
Pacemaker	Si	Si	Si	Si	Si	?	Si	No	Si	Si	Si	Si Candidato
OpenAIS	Si	Si	Si	Si	Si	Si	Si	?	Si	No [11]	Si	No Candidato
Corosync Cluster Engine	Si	Si	Si	Si	Si	?	Si	No	Si	Si	Si	Si Candidato
Linux Virtual Server	Si	Si	Si	Si	Si	Si	Si	?	Si	No	Si	No Candidato
Piranha	Si	Si	Si	Si	Si	?	Si	?	Si	No	Si	No candidato
Ultramonkey	Si	Si	Si	Si	Si	?	?	?	Si	No	Si	No candidato
HAProxy	Si	Si	Si	Si	Si	Si	Si	?	Si	Si	Si	Si Candidato
Pen	Si	Si	Si	?	Si	?	Si	No	Si	No	Si	No candidato
Pound	Si	Si	Si	?	Si	?	Si	?	Si	Si	Si	No Candidato
Zen load balancer	Si	Si	Si	?	Si	?	?	?	Si	Si	Si	No Candidato
Vyatta	Si	Si	Si	Si	Si	?	Si	?	No [12]	Si	Si	No candidato

[11] Se hace la aclaración de que a pesar de ser un proyecto con lanzamiento inferior a la cota determinada, en la página principal del mismo se encuentra un anuncio donde se especifica su abandono al considerar más importante el uso de sus esfuerzos en torno al fortalecimiento del proyecto *Corosync.*

NOMBRE	OBJETIVO DE LA APLICACIÓN	USO DE DESTINO										RESULTADO
	OBJETIVO DE USO MISIÓN CRITICA	LICENCIA APROBADA	CUMPLIMIENTO DE ESTÁNDARES	ADOPTANTES REFERENCIALES	SOPORTE U ORGANIZACIÓN ESTABLE	REVISIONES EFECTUADAS POR TERCEROS	LIBROS	SEGUIMIENTO POR LOS ANALISTAS DEL SECTOR	OBTENCIÓN GRATUITA	ULTIMO LANZAMIENTO RECIENTE	VERSIÓN ESTABLE	
Kimberlite	Si	Si	?	?	?	?	?	?	?	No	Si	No candidato
Keepalived	Si	Si	Si	?	Si	Si	Si	?	Si	Si	Si	Si Candidato
Proxmox VE HA Cluster	Si	Si	Si	?	Si	Si	Si	Si	Si	Si	No[13]	No Candidato
HA Open Solaris Cluster	Si	Si	Si	?	Si	?	?	?	Si	No	Si	No Candidato
HP Service Guard	Si	No	?	?	Si	?	?	?	No	Si	Si	No candidato
Veritas Cluster Server	Si	No	?	?	?	?	?	?	No	Si	Si	No candidato
Red Hat Cluster Suite	Si	Si	Si	Si	?	Si	?	?	Si	Si	Si	Si Candidato
MySQL Cluster	Si	Si	Si	?	Si	Si	Si	Si	Si	Si	Si	Si candidato
Oracle RAC	Si	No	?	?	Si	?	Si	Si	No	Si	Si	No candidato
SUSE Linux Enterprise High Availability Extension	Si	?	Si	Si	Si	?	Si	?	No	Si	Si	No candidato

[12] Según lo recomendado en la página de la comunidad, este producto no está recomendado para Ambientes de Producción, para ello debe ser adquirida la versión paga.
[13] Existen versiones estables del software para trabajo con máquinas virtuales pero no para alta disponibilidad, la versión actual (2.0), la cual contiene mecanismos para alta disponibilidad, a la fecha de la revisión aún se encuentra en su fase Beta.

Es necesario aclarar, que para el caso del proyecto Linux HA, se hizo seguimiento cada uno de los diferentes sub-proyectos que lo conforman, ya que los mismos cuentan con sus propias comunidades, desarrolladores, índices de adopción entre otros elementos que son particulares para cada proyecto.

Como resultado de la fase I se obtiene la siguiente lista de proyectos que son candidatos para una solución tecnológica:

Tabla 3.3 Resultado Fase 1 OpenBRR.

	PROYECTO
1	Cluster Glue
2	Heartbeat
3	Resourse Agent
4	HAProxy
5	Corosync Cluster Engine
6	Pacemaker
7	Keepalived
9	Red Hat Cluster Suite
8	MySQL Cluster

Verificación de Resultados obtenidos:

A pesar de no ser un paso definido en la metodología, según la revisión realizada y los consejos de expertos en el uso de la metodología Open BRR, es recomendable hacer una revisión del resultado del filtro rápido, a fin de dejar un máximo de tres o menos herramientas que continúen con las fases siguientes del proceso, esto tiene como finalidad minimizar los desgastes derivados de la búsqueda de información. Siguiendo esta sugerencia se encontró que los componentes pertenecientes al proyecto Linux HA se encuentran contenidos dentro de los proyectos de mayor envergadura (HA Proxy, Pacemaker, Keepalived y Red Hat Cluster Suite), por esto, se prescinde de continuar haciendo su evaluación, aun así se considera que su paso dentro del filtro rápido fue útil para, de haber sido necesario, eliminar proyectos que pudieran contener como componente el proyecto Open AIS (eliminado durante esa fase), ya que de otra manera esta situación no hubiera sido evaluada en una etapa tan temprana del proceso, ya que esto plantea una evaluación de riesgo profunda que no es lo que se busca dentro del desarrollo de las Fases I de la metodología OpenBRR.

Siguiendo con la revisión de herramientas, en conversaciones con miembros del área de servidores y verificando las herramientas sobre las que se pretende hacer la instalación, se efectuó un listado de criterios adicionales, los cuales fueron:

La solución tecnológica a implementar debe ser compatible con Drupal, un sistema de gestión de contenido modular multipropósito usado en el Portal Web de la Universidad del Cauca para la publicación de artículos, imágenes, u otros archivos y en lo posible su montaje debe ser en la distribución Debian de Linux, a fin de minimizar traumatismos derivados de cambios de tecnologías.

Con los anteriores criterios se encontró que MySQL Cluster no es compatible con Sistema de Gestión de Contenidos Drupal, ya que este último no soporta NDB (un motor de almacenamiento en memoria que ofrece alta disponibilidad y persistencia de datos utilizado por MySQL Cluster) [93], además dentro de la investigación realizada también se encontró que de ser posible dicha instalación resultaría poco práctica, ya que el rendimiento se vería ampliamente afectado [94], por estos motivos MySQL Cluster es descartada. Mientras que Red Hat Cluster Suite aparentemente cumple con todos los criterios para continuar con las siguientes fases, al llevar una revisión más profunda con miras a su implementación fue puesto en manifiesto la gran falta de información de carácter gratuito necesaria para la realización del montaje en Linux Debian, además impactó el poco entusiasmo de la comunidad por este proyecto (dado el bajo número de implementaciones encontradas), adicional a esto después de la formulación de preguntas en diferentes foros y la falta de respuestas de los mismos fue notorio que el soporte también iba a ser un problema ya que no se dispone de recursos económicos para adquirir el brindado directamente por Red Hat Enterprise, por todo lo anterior fue descartada como posible parte de la solución tecnológica dadas las condiciones específicas de este proyecto haciendo que la lista tenga una considerable reducción quedando:

Tabla 3.4 Herramientas candidatas para evaluar las siguientes fases

HERRAMIENTAS PARA EVALUAR LAS SIGUIENTES FASES
HA Proxy
Keepalived
Pacemaker

3.3. DISEÑO DE LA SOLUCIÓN TECNOLÓGICA

Una vez que se ha determinado los problemas especificados en el Capítulo 2 del presente documento e identificadas las herramientas posibles a emplear, se procede a especificar la posible(s) solución(es) tecnológica(s).

3.3.1. Diseño de la Solución Tecnológica para el Portal Web

En este apartado se presenta el diseño de la arquitectura que ofrecerá el soporte al portal del sitio web que posee la Universidad del Cauca, esta se encuentra relacionada con la publicación de contenidos de comunicados, resoluciones, acuerdos, circulares, eventos académicos, científicos y culturales constituyéndose en uno de los medios de comunicación tanto al interior como exterior de la institución por lo que demanda un consumo importante de recursos, tales como: establecimiento de conexiones, alto tráfico generado tanto desde la intranet como desde internet, consultas , peticiones y respuestas realizadas a los servidores Web, y al motor de Base de datos que permite el alojamiento de los contenidos gestionados por Drupal. De la relación de herramientas (Ver tabla 3.4) las diversas posibilidades de construcción de soluciones, y la evaluación realizada en el capítulo 2 donde se determinó dos capas problema, para las cuales se propone lo siguiente:

- **Capa balanceo de cargas**:

Esta capa puede emplearse e implementarse a nivel de balanceadores de carga hardware el cual puede ofrecer enormes ventajas para este fin, pero para este proyecto no se considera principalmente por su alto costo de adquisición, por consiguiente de las herramientas listadas en la tabla 3.4 es posible determinar dos diferentes soluciones, estas son:

1. **Solución tecnológica 1:**

   El balanceo de cargas se hace a nivel software mediante la instalación, configuración e implementación de HAproxy y Keepalived sobre servidores con distribución Linux.

   Se tienen dos balanceadores de carga (B1 y B2) configurados en modo activo-pasivo, donde se define un servidor maestro y otro esclavo comunicados mediante Keepalived. Keepalived hace uso de un *heartbeat* para mantener informados a los nodos de esta capa de algún tipo de caída del nodo principal del balanceador, si esto sucede se procede a conmutar la IP virtual del servicio al nodo definido como esclavo para tener así una capa redundante para mantener ofreciendo el balanceo de cargas y permitir tener continuidad del servicio con una pérdida de tiempo mínima entre la transición de nodos.

   HAProxy se configura además para que tenga comunicación con los servidores Web (W1 y W2), esto debido a que los nodos del balanceador de carga deben saber cuáles servidores Web tiene a disposición y están disponibles en ese instante de tiempo para atender a los diversos usuarios que quieran acceder a la aplicación. La distribución de carga soportada por cada uno de los servidores Web dependerá de la configuración del modo en que se hará el balanceo de carga (*round robin*, *round robin* ponderado, servidor con menos conexiones activas, servidor con menos conexiones activas ponderado, entre otros algoritmos para balanceo)

2. **Solución tecnológica 2:**

   Está solución tecnológica es muy parecida a la primera, el balanceo de cargas se hace a nivel software mediante la instalación, configuración e implementación de LVS y Keepalived, cabe aclarar que LVS es un paquete integrado a Keepalived, y que esta instalación se hará sobre servidores con distribución Linux Debían, siendo válido aquí también para el balanceo la configuración activo-pasivo y el modo de funcionamiento de Keepalived descrito para la solución tecnológica anterior

- **Base de datos**:

Para este caso se encontró que las soluciones listadas son útiles para alta disponibilidad y balanceo de carga de servidores web pero necesitaban de configuraciones especiales para la alta disponibilidad de bases de datos, para lograrlo necesitaban hacer uso de una funcionalidad de MySQL server, la replicación, la cual se encuentra disponible desde la versión 3.5.5, [95], y permite realizar una réplica del contenido en la base de datos, además de esto es posible realizar una configuración de tipo maestro-maestro, adicional a lo anterior, se hace necesaria la implementación de una dirección IP Virtual que es a la cual los servidores se comunicarán por

defecto, esto a fin de que si se produce un fallo, el cambio de servidor de base de datos sea transparente para las capas superiores, así se llega a dos posibles soluciones:

1. Configurando e instalado Keepalived, y estableciendo para efectos de balanceo de carga un nodo que actuara como primario y otro como secundario, Keepalived hace uso de un *heartbeat* para mantener informados a los nodos de esta capa de algún tipo de caída del nodo principal, si esto sucede se procede a pasar la IP virtual del servicio al nodo definido como esclavo para tener así mediante el uso de la redundancia y seguir ofreciendo los contenidos alojados y con esto permitir tener continuidad del servicio con pérdida de tiempo mínima entre la transición de nodos.
2. La segunda opción es la configuración y puesta en marcha de Pacemaker, donde Pacemaker para la gestión de recursos en alta disponibilidad, Pacemaker hace uso de Corosync como capa de comunicación, la configuración a implementar va ser del tipo activo-pasivo, donde el nodo que actué como maestro es el encargado de hacer la publicación de la IP Virtual y el nodo que actúa como esclavo entrará en funcionamiento sólo en caso de producirse una falla.

Es importante aclarar que MySQL con replicación y MySQL cluster no son lo mismo, la diferencia radica en que la primera aun cuando se encuentre en replicación maestro-maestro funciona de manera asíncrona mientras que la segunda trabaja de manera síncrona [96], esta funcionalidad de MySQL Server no fue evaluada dentro del filtro rápido ya que se trata de una funcionalidad y no de una herramienta como tal.

Dicho todo lo anterior se tiene que las posibles soluciones y escenarios de prueba para las mismas son:

A. Keepalived + MySQL con replicación, donde el escenario de pruebas se observa en la Figura 3.1.

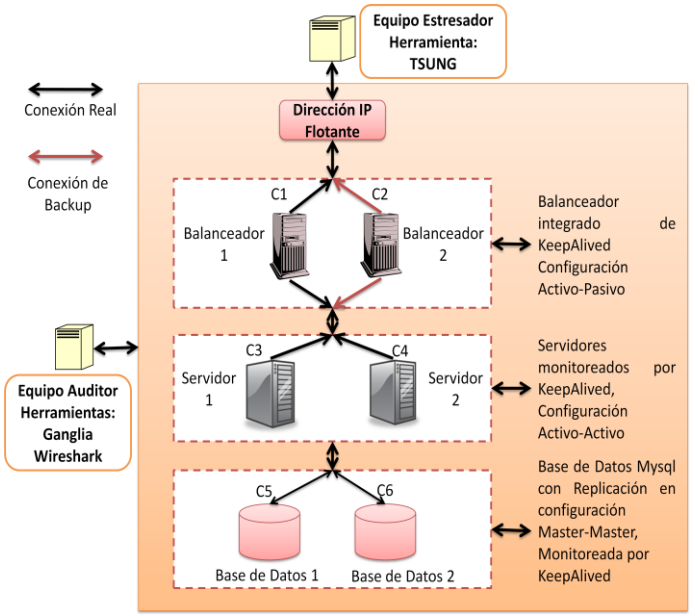

Figura 3.1 Keepalived + MySQL con replicación

B. HA Proxy + Keepalived + MySQL con replicación, el escenario de pruebas se observa en la Figura 3.2.

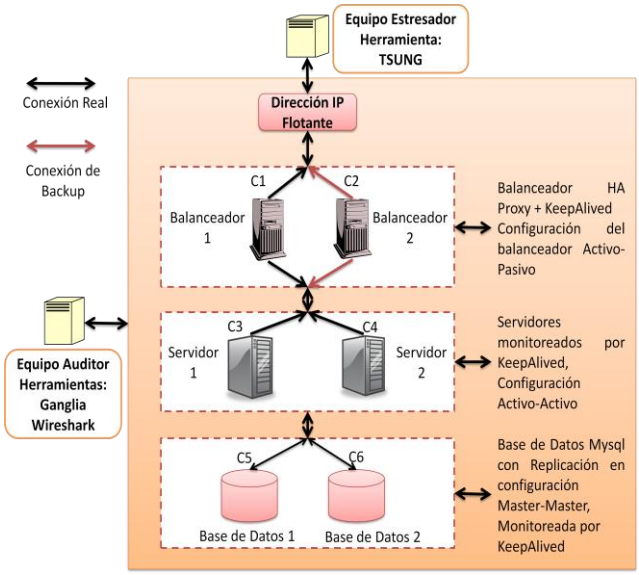

Figura 3.2 Keepalived + HA Proxy + MySQL con replicación

C. HA Proxy + Keepalived + Pacemaker + MySQL con replicación, el escenario de pruebas se observa en la Figura 3.3.

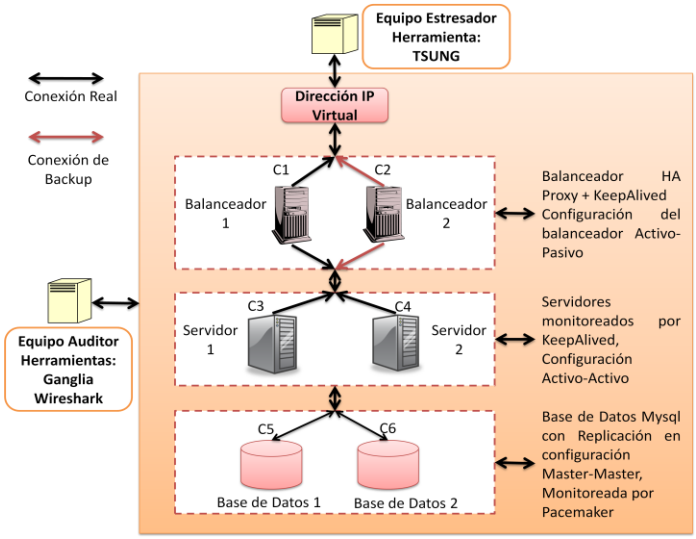

Figura 3.3 HA Proxy + Keepalived + Pacemaker + MySQL con replicación.

3.3.2. Diseño de la Solución Tecnológica para SIMCA

En este apartado se presenta el diseño de la arquitectura que ofrecerá el soporte a una de las aplicaciones más críticas que tiene la Universidad del Cauca, esta se encuentra relacionada con los procesos a nivel académico y docente, lo cual demanda un alto consumo de recursos, tales como: el establecimiento de conexiones concurrentes, alto tráfico generado tanto desde la intranet como

desde Internet, consultas , peticiones y respuestas realizadas a los servidores Web, servidores de aplicaciones y a los motores de Base de datos entre otros elementos involucrados en este proceso.

A. Arquitectura detallada

Arquitectura Física:

Para el diseño de la arquitectura a nivel físico y funcional, se tuvieron las siguientes consideraciones:

Se diseña la solución tecnológica de la arquitectura sólo para el despliegue de la aplicación SIMCA.

Existe ya una infraestructura de red y demás elementos tanto internos como externos para interconectar esta solución (Switch's de distribución, de acceso, de *core*, *firewall*, servidores de resolución de nombres, entre otros).

Las condiciones para el despliegue de la aplicación son las siguientes:

- Servidor de Aplicaciones Jboss.
- Motor de Base de datos Oracle 11g.
- Almacenamiento compartido desde Storage, NAS o alguna otra área de almacenamiento que sea común para los servidores en clúster implementados.

Además, según lo concluido en el capítulo 2, es posible determinar que la solución tecnológica debe estar enfocada en brindar solución a los problemas hallados en las capas de balanceo de carga y persistencia.

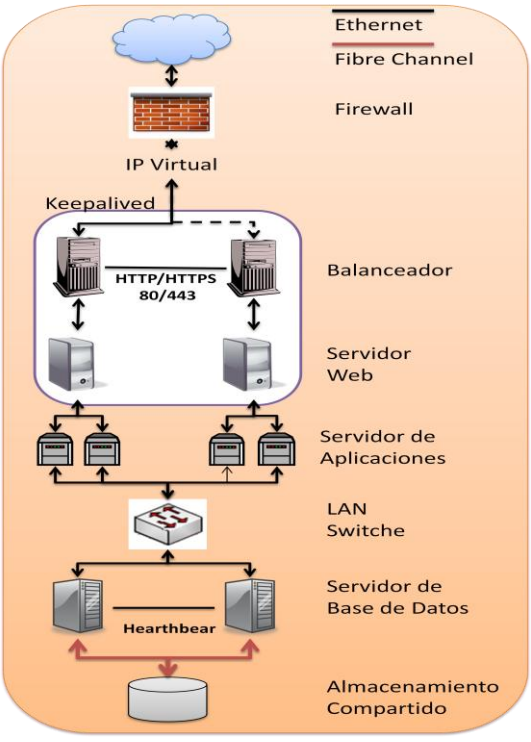

Figura 3.4 Arquitectura Física.

Las soluciones propuestas son:

- **Capa balanceo de cargas**:

Al igual que para el portal Web, esta capa puede emplearse e implementarse a nivel de balanceadores de carga hardware, pero para este proyecto no se considera principalmente por su alto costo de adquisición, por consiguiente de las herramientas listadas en la tabla 3.4 es posible determinar que la solución apropiada es la constituida por HAProxy + Keepalived, esto a razón de que según la revisión realizada y los consejos de expertos en su uso, el balanceador integrado con Keepalived (LVS), a diferencia de HAproxy, no soporta "*Frontend"* para Jboss de manera nativa por lo que se descarta esta opción como posible solución.

- **Capa de Persistencia**

Es en esta capa donde los datos de la aplicación se almacenan, su función es ofrecer servicios de persistencia y recuperación de información a las capas superiores, para el caso del presente proyecto, en esta capa, se cuenta con la base de datos Oracle, es por ello que fruto de la investigación realizada se descubrió que un posible problema con dicha capa radica en la falta de redundancia de servidores de base de datos, ya que esta se constituye por tanto en un punto único de fallo o *SPOF (Single Point Of Failure)* para todo el sistema, la posible solución tecnológica consiste en la puesta en marcha de un Oracle RAC (*Real Aplication Cluster*), la cual es suministrada por la empresa Oracle y es de carácter privativo no gratuito; no fue posible encontrar dentro del software libre un desarrollo o proyecto que satisfaga la alta disponibilidad para bases de datos Oracle, adicional a ello la recomendación generalizada de varios profesionales consultados fue la utilización del RAC en entornos de producción, así mismo, dentro de la investigación realizada se evidenció la existencia de equipos fabricados que son certificados por la propia empresa Oracle para ser empleados en el montaje de Oracle RAC, a pesar de ello se intentó implementarlo pero se fracasó debido a que Oracle RAC requiere una amplia capacidad de recursos de hardware, por ello su implementación va más allá de los alcances del presente proyecto.

3.4. DEFINICIÓN DE TIPOS PRUEBAS Y HERRAMIENTAS DE MONITOREO

3.4.1. Pruebas para determinar la funcionalidad

El proceso de prueba del software puede definirse como: "*la verificación dinámica del comportamiento del software a partir de un conjunto finito de casos de prueba"* [97], para probar si la solución tecnológica planteada cumple con las características en cuanto a alta disponibilidad y balanceo de carga se establecieron las siguientes pruebas:

A. Pruebas para balanceo de carga:

Para la revisión del balanceo de carga se realizaron pruebas de Valores Frontera, con las cuales, se buscan probar situaciones extremas o inusuales que la solución tecnológica debe ser capaz de manejar [98].

Para esto se efectúan pruebas de estrés cuyo objetivo es simular condiciones extremas, para ello se hace una elevación sistemática del número de usuarios que efectúan solicitudes y se verifica que el uso de la CPU en los nodos sea el mismo, también se pretende determinar el número

máximo de usuarios que puede contener el sistema antes de experimentar una caída o de iniciar el negado de peticiones nuevas.

Al incrementar el número de usuarios que efectúan peticiones se pretende hallar el valor máximo de tráfico posible.

B. Pruebas para alta disponibilidad:

Para establecer cuáles son las pruebas oportunas para alta disponibilidad, es necesario referirnos a los conceptos básicos de alta disponibilidad, donde partiendo de la ecuación 2 (Fórmula del cálculo de la disponibilidad de los servicios), se observa que los índices de disponibilidad están directamente ligados a los tiempos de caída, es por ello que es imperante conocer los tiempos de respuesta que pueda tener una herramienta, para tal fin se hará la simulación de caídas en diferentes momentos del tráfico y se buscara mediante herramientas de monitoreo evaluar el tiempo en que el servicio no se encontró disponible.

3.4.2. Herramientas de monitoreo y de pruebas

1. **Ganglia:** Es un sistema escalable y distribuido para el monitoreo de clústeres y Grids computacionales en tiempo real. Es una implementación robusta que ha sido adaptada a muchos sistemas operativos y distintos arquitecturas de computadores, y es actualmente usada en miles de clústeres alrededor del mundo como universidades, laboratorios de investigación comerciales y gubernamentales.

   Fue inicialmente desarrollado por la Universidad de Berkeley en el departamento de ciencias computacionales para enlazar los clústeres del campus, actualmente está a cargo de SourceForge.

   Está basado en un esquema jerárquico de clústeres y es configurado mediante archivos XML y XDR en cada nodo que permite tener extensibilidad y portabilidad. Es completamente Open-Source y no contiene ningún componente propietario. Ganglia enlaza líneas de clústeres y computación distribuida por lo que se conoce como “Clúster to Clúster” [99]

2. **Wireshark:** Es un analizador de protocolos *open-source* diseñado por Gerald Combs y que actualmente está disponible para plataformas Windows y Unix. El principal objetivo de Wireshark es el análisis de tráfico además de ser una excelente aplicación didáctica para el estudio de las comunicaciones y para la resolución de problemas de red. Tiene implementado una amplia gama de filtros que facilitan la definición de criterios de búsqueda para los más de 1100 protocolos soportados actualmente (versión 1.4.3); y todo ello por medio de una interfaz sencilla e intuitiva que permite desglosar por capas cada uno de los paquetes capturados. Wireshark “entiende” la estructura de los protocolos, permitiendo la visualización de los campos de cada una de las cabeceras y capas que componen los paquetes monitorizados, proporcionando un gran abanico de posibilidades al administrador de redes a la hora de abordar ciertas tareas en el análisis de tráfico. [100]

3. **Tsung:** Es una herramienta de prueba de carga multiprotocolo y distribuida, fue desarrollada en 2001 por Nicolas Niclausse. Se emplea para testear: HTTP, WebDAV, LDAP, MySQL,

PostgreSQL, SOAP y los servidores Jabber (soporta SSL). Tsung es software libre y se distribuye bajo la licencia GPLV2. El objetivo es la simulación del comportamiento de cientos de miles de usuarios concurrentes usando un archivo XML, muestra muchas medidas en tiempo real: tiempos de reacción incluyendo, uso de la CPU y de la memoria. También puede funcionar en cluster.

3.5 ELABORACIÓN DE PRUEBAS Y DETERMINACIÓN DE PUNTAJES PARA LAS DIFERENTES SOLUCIONES TECNOLÓGICAS

Una vez establecidas las posibles soluciones es necesario proceder con el desarrollo de las fases 3 y 4 de la metodología Open BRR, recolección de datos y procesamiento, y selección del aplicativo respectivamente, para esto con las posibles soluciones ya establecidas, es necesario continuar con la evaluación profunda de la solución, con este fin van a ser utilizados el segundo conjunto de criterios establecidos en el capítulo 1 del presente proyecto en su sección 1.5.2.

Primero se procederá con la recolección de los datos correspondientes a la funcionalidad, pero a fin de ser colocados en la plantilla del método de evaluación OpenBRR, es necesaria la realización de pruebas, ya que las métricas elegidas para la funcionalidad dependen del tipo de máquinas sobre el que esté trabajando, resultando únicas para cada tipo de escenario de trabajo y por lo tanto no es posible encontrarlas como parte de la documentación de las herramientas, para ello se procedió de la siguiente forma:

3.5.1. Elaboración de las pruebas de funcionalidad para la solución tecnológica del portal Web

Debido a la complejidad y extensión de cada uno de las capturas de pantalla para este tipo de procedimiento se optó por dejar las mismas en el anexo D y presentar aquí sólo las tablas donde se resumen los resultados obtenidos.

1. **Escenario descrito en la Figura 3.2. (Keepalived + MySQL replicación.)**
   A. **PRUEBAS DE BALANCEO DE CARGA:**

Tabla 3.5 Resumen de las pruebas de balanceo de carga para la solución tecnológica Keepalived + MySQL.

EQUIPO DE PRUEBA	PUNTO DE MEDICION	NÚMERO DE USUARIOS										
		20	50	100	500	1000	1500	2000	2500	2800	3000	3500
Balanceador 1	USO DE LA CPU	3%	2%	3%	3%	8%	5%	10%	20%	30%	35%	45%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si, un error 503.
Balanceador 2	USO DE LA CPU	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si, un error 503.
Servidor 1	USO DE LA CPU	3%	3%	4%	21%	37%	45%	70%	78%	90%	96%	100%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si, un error 503.
Servidor 2	USO DE LA CPU	3%	3%	4%	20,5%	37%	48%	70%	80%	88%	89%	92%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si, un error 503.
Base de datos 1	USO DE LA CPU	2%	2%	2%	4%	12%	10%	30%	30%	40%	40%	42%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si, un error 503.
Base de datos 2	USO DE LA CPU	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si, un error 503.

De la anterior tabla surgen las siguientes conclusiones:

- El mayor número de personas que el sistema soporto antes de una caída fue de 3000 usuarios.
- La caída se presentó con 3500 usuarios, aun así es destacable el hecho que la caída se presentó en los servidores, ello indica que un aumento de los mismos lograría un aumento en el tráfico que es posible manejar.

- Durante la caída fue posible observar que inmediatamente el servidor supero sus límites de procesamiento el balanceador automáticamente lo elimino del *cluster* y unos segundos después lo volvió a integrar cuando detecto que era posible su reingreso.
- Dado que los servidores se constituyen en el punto de fallo se evaluaron los mismos para determinar los niveles de tráfico adecuados para las siguiente fase de pruebas, y después de una evaluación de los niveles de tráfico y uso de los equipos que componen se determinó como un tráfico alto a 2800 usuarios ya que es allí donde se tienen niveles cercanos al 80% de uso de los recursos, niveles que ya resultan altos, como tráfico medio se determinó 1500 usuarios esto obedece al hecho de que es en esta valor donde se aprecia un uso del 50%.
- El balanceo de carga se llevó a cabalidad, ello es observable en el hecho de que los niveles de uso de recursos fue el mismo para los elementos entre los cuales se pretendía dividir las solicitudes.

B. PRUEBAS PARA ALTA DISPONIBILIDAD:

Tabla 3.6 Tiempo de Caída según el tipo de tráfico para la solución tecnológica Keepalived + MySQL

ELEMENTO SOBRE EL QUE SE SIMULARA LA CAIDA	TIEMPO DE CAIDA SEGUN EL TIPO DE TRAFICO (en segundos)		
	TRAFICO ALTO	TRAFICO MEDIO	TRAFICO BAJO
Balanceador Principal	4	3	4
Nodo de Servidor Web	2	4	4
Base de Datos Principal	2	3	3

2. Escenario descrito en la Figura 3.3. (HA Proxy+ Keepalived + MySQL replicación.)

Tabla 3.7 Resumen de las pruebas de balanceo de carga para la solución tecnológica HA Proxy +Keepalived + MySQL.

EQUIPO DE PRUEBA	PUNTO DE MEDICION	NÚMERO DE USUARIOS										
		20	50	100	500	1000	1500	2000	2500	2800	3000	3500
Balanceador 1	USO DE LA CPU	5%	7%	7%	7%	14%	20%	30%	38%	39%	40%	45%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si
Balanceador 2	USO DE LA CPU	7%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si
Servidor 1	USO DE LA CPU	3%	3%	3%	17%	28%	38%	50%	57%	60%	78%	95%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si
Servidor 2	USO DE LA CPU	3%	3%	3%	18%	30%	38%	51%	58%	60%	78%	100%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si
Base de datos 1	USO DE LA CPU	2%	2%	2%	7%	15%	23%	25%	40%	40%	42%	45%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si
Base de datos 2	USO DE LA CPU	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si

Igual que en el escenario anterior las conclusiones de este son similares, con el adicional de que se percibió que las caídas ocurrieron en un tiempo mucho más amplio que lo ocurrido con las solución tecnológica anterior, haciendo que el sistema perdiera menos usuarios, según los reportes originados por la herramienta Tsung, convirtiéndose esto en un punto extra para la calificación de funcionalidad.

B. PRUEBAS PARA ALTA DISPONIBILIDAD:

Tabla 3.8 Resumen de los Tiempos de Caída para Keepalived + MySQL.

ELEMENTO SOBRE EL QUE SE SIMULARA LA CAIDA	TIEMPO DE CAIDA SEGUN EL TIPO DE TRAFICO (en segundos)		
	TRAFICO ALTO	TRAFICO MEDIO	TRAFICO BAJO
Balanceador Principal	4	2	4
Nodo de Servidor Web	3	1	2
Base de Datos Principal	2	2	4

3. Escenario descrito en la Figura 3.3. (HA Proxy+ Keepalived + Pacemaker + MySQL replicación.)

Tabla 3.9 Resumen de las pruebas de balanceo de carga para la solución tecnológica HA Proxy +Keepalived + Pacemaker + MySQL.

EQUIPO DE PRUEBA	PUNTO DE MEDICION	NÚMERO DE USUARIOS										
		20	50	100	500	1000	1500	2000	2500	2800	3000	3500
Balanceador 1	USO DE LA CPU	5%	7%	7%	7%	14%	20%	30%	38%	39%	40%	45%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si
Balanceador 2	USO DE LA CPU	7%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si
Servidor 1	USO DE LA CPU	3%	3%	3%	17%	28%	38%	50%	57%	60%	78%	95%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si
Servidor 2	USO DE LA CPU	3%	3%	3%	18,0%	30%	38%	51%	58%	60%	78%	100%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si
Base de datos 1	USO DE LA CPU	2%	2%	2%	7%	15%	23%	25%	40%	40%	42%	45%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si
Base de datos 2	USO DE LA CPU	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200	200	200	200	200, 503
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No	No	No	No	Si

Igual que para los escenarios anteriores las conclusiones de este son similares, con el adicional de que se percibió mayor estabilidad en la base de datos, y una mejor gestión de la misma.

C. PRUEBAS PARA ALTA DISPONIBILIDAD:

Tabla 3.10 Resumen de los Tiempos de Caída para HA Proxy+ Keepalived + Pacemaker + MySQL replicación.

ELEMENTO SOBRE EL QUE SE SIMULARA LA CAIDA	TIEMPO DE CAIDA SEGUN EL TIPO DE TRAFICO (en segundos)		
	TRAFICO ALTO	TRAFICO MEDIO	TRAFICO BAJO
Balanceador Principal	4	2	4
Nodo de Servidor Web	3	1	2
Base de Datos Principal	2	2	4

Después de la realización de las pruebas anteriormente mencionadas en cuanto a alta disponibilidad es posible constatar que de presentarse una caída por fallo de comunicación el nodo comprometido seria inmediatamente retirado del *cluster*, en cuanto a los cuelgues también fue posible constatar que al presentarse usos exagerados de la CPU (ver anexo D, Prueba 11), el balanceador inmediatamente retiró el nodo comprometido y lo volvió a ingresar pasados unos segundos, también fue posible constatar que el balanceo de carga se estaba haciendo de manera correcta, durante todas las pruebas, demostrándose así que la simulación de una caída en un nodo del portal web tiene el mismo impacto que de tratarse del otro, en el balanceo de carga se constató que el balanceador secundario se encontraba a la espera razón por la cual hacer la prueba de bajarlo resulta sin influencia para el servicio prestado, por esto a fin de simplificar las pruebas estas fueron obviadas, concluyéndose las pruebas realizadas son una buena aproximación a los casos reales que se pueden encontrar, así como también que cualquiera de las soluciones tecnológicas en cuanto a funcionamiento dieron resultados positivos.

3.5.2. Definición de Puntajes para las soluciones tecnológicas del portal Web

Como resultado de lo anterior es posible hacer el llenado de los criterios listados en la tabla 1.8 y con ello establecer lo valores de puntuación de la categoría funcionalidad quedando la tabla 3.12.

Tabla 3.11 Puntajes definitivos de Funcionalidad.

FUNCIONALIDAD DE LA SOLUCIÓN TECNOLÓGICA	PUNTAJE DEFINITIVO
Keepalived + MySQL con replicación	**5**
Keepalived + HA Proxy + MySQL con replicación	**5**
Keepalived + HA Proxy + Pacemaker + MySQL con replicación	**5**

Continuando con la evaluación, para cada una de las herramientas anteriormente listadas, se hace una recolección de datos a fin de recoger la información para los criterios listados en la tabla 1.9 y con estos datos llenar las hojas de cálculo (Herramienta con la que cuenta OpenBRR para realizar la evaluación), estas van a ser llenados mediante la consulta de fuentes tales como página oficial, libros especializados en el tema, foros, listas de correo, etc., como resultado de lo anterior se tiene la tabla individual para cada herramienta, las cuales se encuentran en el anexo D, es importante

resaltar que estas hojas de cálculo son entregadas dentro del marco de la metodología Open BRR, a continuación se presenta un resumen de los puntajes obtenidos:

Tabla 3.12 Resumen de los puntajes obtenidos.

PROYECTO	CRITERIOS						Puntaje acumulativo parcial
	Seguridad	Comunidad	Soporte	Documentación	Calidad	Escalabilidad	
HAProxy	5.0	3.5	3.5	4.5	3.6	5	3.105
Keepalived	3.60	1.50	3	4.5	4	5	2.37
Pacemaker	5	2,5	4.5	3,5	3.3	5	3.59

Puntajes definitivos para las Soluciones tecnológicas propuestas para el Portal Web

Para los puntajes definitivos es necesario hacer una ponderación de los valores obtenidos en la tabla 3.11 ya que cada uno constituye un elemento de la solución, esto se hace con el fin de encontrar el valor para la solución tecnológica completa y no sólo para los proyectos individuales que la conforman, asi es posible obtener la tabla 3.13 que contiene los valores definitivos de puntuación para cada una de las soluciones.

Tabla 3.13 Puntajes definitivos para la solución tecnológica

SOLUCIÓN TECNOLÓGICA	PUNTAJE TOTAL
Keepalived + MySQL con replicación	**3.62**
Keepalived + HA Proxy + MySQL con replicación	**3.94**
Keepalived + HA Proxy + Pacemaker + MySQL con replicación	**4.25**

Así se llega a la conclusión que la mejor opción de solución tecnológica es la constituida por HA Proxy, Keepalived y Pacemaker, además de que se puede esperar un poco más de soporte ya que existen múltiples ejemplos de esta configuración, se notó en las pruebas un mejor manejo de recursos de HA Proxy haciendo que el sistema soporte una cantidad mayor de usuarios.

3.5.3. Elaboración de las pruebas diseñadas para la solución tecnológica de SIMCA

Para esta sección se determinó que el servidor de aplicación Jboss está capacitado para publicar la página web aun cuando no tenga comunicación con la base de datos así que dado el funcionamiento de las herramientas de monitoreo y estrés utilizadas se estableció que la mejor forma de realizarlas era separando las pruebas del balanceador de las pruebas realizadas para la capa de persistencia.

Pruebas para la capa de Balanceo de Carga: Para este caso se implementó el escenario mostrado en la figura 3.5.

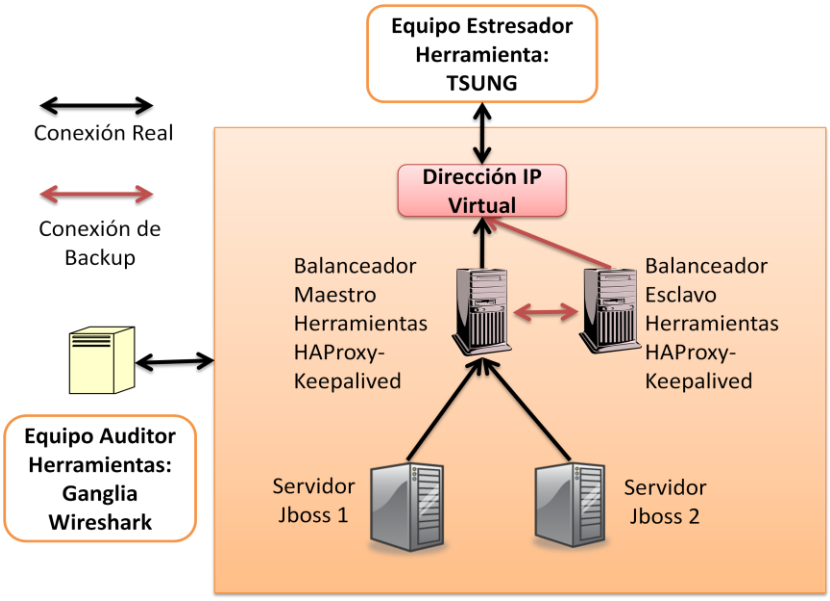

Figura 3.5 Escenario de Pruebas para la capa de balanceo de la solución tecnológica para SIMCA

A. PRUEBAS DE BALANCEO DE CARGA:

Tabla 3.14 Resumen de las pruebas de balanceo de carga para la solución tecnológica para SIMCA

EQUIPO DE PRUEBA	PUNTO DE MEDICION	NÚMERO DE USUARIOS						
		100	500	1000	2000	3000	4000	5000
Balanceador 1	USO DE LA CPU	5%	7%	7%	7%	14%	20%	30%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No
Balanceador 2	USO DE LA CPU	7%	5%	5%	5%	5%	5%	5%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No
Servidor 1	USO DE LA CPU	3%	3%	3%	3%	3%	3%	3%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No
Servidor 2	USO DE LA CPU	3%	3%	3%	3,0%	3%	3%	3%
	TIPO DE RESPUESTA HTTP	200	200	200	200	200	200	200
	¿El PORTAL PRESENTA CAIDAS?	No	No	No	No	No	No	No

Debido a limitantes en las herramientas no fue posible continuar elevando el número de usuarios pero fue posible determinar su estabilidad y que su aumento de seguir como iba podía llegar a los 10000 usuarios, aun así se estableció como un tráfico alto 5000 usuarios que corresponde a casi el 50% de la población estudiantil ingresando al mismo tiempo, cosa que dentro de las indagaciones realizadas no es un escenario muy factible dada la división en franjas horarias para determinar el ingreso al sitio en su periodo más alto de ingreso (Matriculas Académicas), como trafico medio se establece 2500 usuarios y como bajo 50 usuarios.

Tabla 3.15 Resumen de los puntajes obtenidos

ELEMENTO SOBRE EL QUE SE SIMULARA LA CAIDA	TIEMPO DE CAIDA SEGUN EL TIPO DE TRAFICO (en segundos)		
	TRAFICO ALTO	TRAFICO MEDIO	TRAFICO BAJO
Balanceador Principal	4	2	4
Nodo Jboss	3	1	2

3.5.4. Definición del Puntaje para la solución tecnológicas de SIMCA en su capa de Balanceo de Carga

Con los parámetros anteriormente listados y al igual que para la solución anterior se llenan los parámetros de funcionalidad obteniéndose la tabla 3.16.

Tabla 3.16 Puntaje de funcionalidad para la solución tecnológica dada para SIMCA

FUNCIONALIDAD DE LA SOLUCIÓN TECNOLÓGICA	PUNTAJE DEFINITIVO
Keepalived + HA Proxy	**5**

Con estos parámetros se procedió a llenar la hoja de evaluación para las herramientas, la cual da como resultado la tabla 3.17:

Tabla 3.17 Puntajes por categoría para los elementos de la solución dada para SIMCA

PROYECTO	CRITERIOS						Puntaje acumulativo parcial
	Seguridad	Comunidad	Soporte	Documentación	Calidad	Escalabilidad	
HAProxy	5.0	3.5	3.5	4.5	3.2	5	3.105
Keepalived	3.60	1.50	3	4.5	4	5	2.37

Siguiendo el mismo esquema explicado para la solución anterior se establece que el puntaje definitivo para esta se especifica en la tabla 3.18.

Tabla 3.18 Puntaje total de la solución dada para SIMCA

SOLUCIÓN TECNOLÓGICA	PUNTAJE TOTAL
Keepalived + HA Proxy + MySQL con replicación	**3.94**

Pruebas para la capa de Persistencia: El escenario para las pruebas se ilustra en la figura 3.6., se puede apreciar como la capa de Persistencia consta de una arquitectura basada en un clúster activo – activo de dos nodos, en los que corre el Sistema Operativo Oracle Enterprise Linux (OEL) Versión 5 Update 8 para x86_64, Oracle RAC 11g Release 2 para Linux x86_64 y 2,0 ASMLib, se tiene un tercer nodo empleado como un disco compartido de almacenamiento basado en tecnología iSCSI en el que corre Openfiler Release 2.99 x86_64, este nodo es conocido como el servidor de almacenamiento de red. Los nodos del cluster están conectados por una red Pública que permite el acceso a los usuarios y una red privada para gestión propia del Cluster y recuperación de datos. Las pruebas son clasificadas en Failover en la Base de Datos y Failover en el Hardware.

En la pruebas de failover en la Base de Datos se tienen failover ante eventos de fallas de una instancia de la base de datos, failover ante evento de falla en el servidor, balanceo de cargas para numerosas conexiones y pruebas de estabilidad ante eventos de falla en los procesos propios de la tecnología de Oracle. En las pruebas failover en el Hardware se tienen los eventos de falla que puedan presentarse en las tarjetas de red y equipos o elementos de comunicaciones de la red.

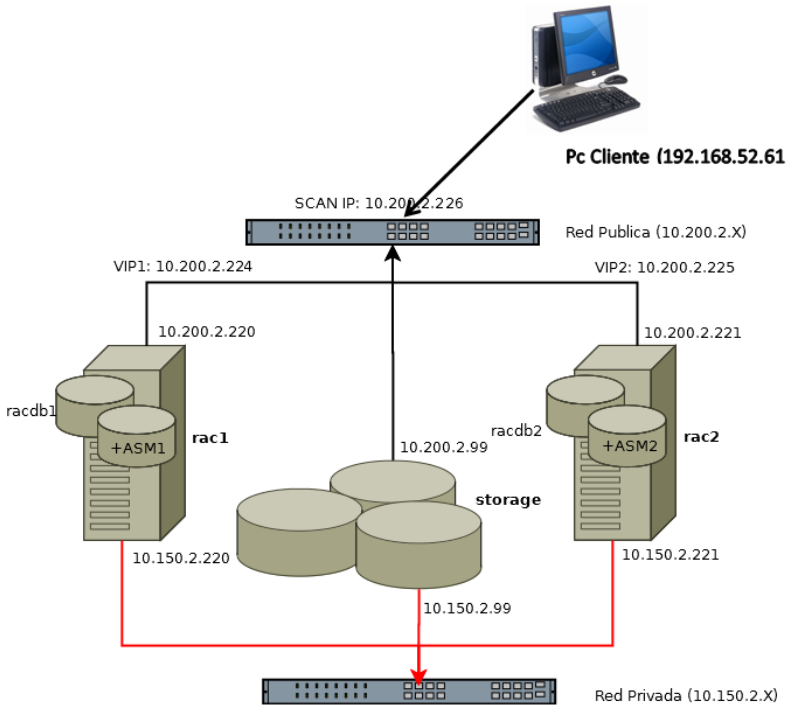

Figura 3.6 Escenario de Pruebas para la capa de persistencia de la solución tecnológica para SIMCA

A. PRUEBAS FAILOVER EN LA BASE DE DATOS

Prueba Nro. 1

Objetivo: Probar el failover o conmutación por fallo de una conexión de un usuario en el evento de una falla en una instancia

Tabla 3.19 Descripción de la Prueba Nro.1

Acción	Expectativa	Observación	Resultado

1. Crear un usuario HR y una tabla perteneciente al mismo usuario HR.
 	
SQL> create user HR identified by HR default tablespace USERS temporary tablespace TEMP;
SQL> grant dba, resource, connect to HR;
SQL> create table HR (Col1 number);
 	
2. Insertar cerca de 100,000 líneas de datos en la tabla
 	
SQL> declare
SQL> x number;
SQL> begin
SQL> for x in 1..100000 loop
SQL> insert into test values (x);
SQL> end loop;
SQL> end;
SQL> /
 	
3. Desde el Pc Cliente conectarse a la base de datos a través de sqlplus:
$ sqlplus HR/HR@RACDB
4. Verificar a cual instancia está conectado el usuario HR:
SQL> select instance_name from v$instance;
5. Conectarse por SSH al nodo o instancia al cual el usuario HR está conectado.
C:> ssh root@rac1 o ssh root@rac2
$ ORACLE_SID=RACDB1/2; export ORACLE_SID
$ sqlplus / as sysdba
 	
6. Ejecutar una sentencia select por parte del usuario HR e inmediatamente apague la instancia a la cual HR está conectado
HR: SQL> select * from HR;
root@rac1o2: SQL> shutdown immediate;
	La conexión de SQL debe ser capaz de cambiar a otra instancia y aun así continuar con la consulta de selección (select).
	1. La conexión de sqlplus desde el cliente a la base de datos aún está disponible
	
	2. La sentencia SELECT aún es procesada
		La consulta select aún es procesada mientras que la instancia está apagada	Pasa

Prueba Nro. 2

Objetivo: Probar el failover o conmutación por fallo completo de una sesión en el evento de un nodo como el apagado repentino.

Tabla 3.20 Descripción de la Prueba Nro.2

Acción	Expectativa	Observación	Resultado
1. Desde el Pc Cliente conectarse a la base de datos a través de sqlplus: $ sqlplus HR/HR@RACDB 2. Verificar a cual instancia está conectado el usuario HR: SQL> select instance_name from v$instance; 3. Ejecutar una sentencia select por parte del usuario HR e inmediatamente el nodo o maquina cuya instancia este el usuario HR conectado HR: SQL> select * from test;	La sesión SQL no debería verse afectada y el failover debería efectuarse al nodo siguiente sin ninguna interrupción	La consulta select aún es procesada mientras que la maquina está apagada	Pasa

Acción	Expectativa	Observación	Resultado
root@rac1o2 #: poweroff;			

Prueba Nro. 3

Objetivo: Probar el balanceo de carga de Oracle RAC bajo numerosas conexiones.

Tabla 3.21 Descripción de la Prueba Nro.3

Acción	Expectativa	Observación	Resultado
1. Crear un script en Linux que simule una conexión a la base datos y efectué una consulta e imprima la instancia en la que se realizó la conexión. ********** script consulta.sh********** #!/bin/bash For j in $(seq 1 20); do echo "Conexion Nro. $j" sqlplus –S HR/HR@RACDB @verinstancia.sql done **********fin script consulta.sh********** **********verinstancia.sql********** SELECT instance_name FROM v$instance; exit; **********fin verinstancia.sql ********** 2. Correr el script unas 20 o 300 veces en una sola vez 3. Comprobar que este el número de conexiones según las veces que se ejecutó el script y cuales instancias están las conexiones presente.	Leer el nombre de la instancia como se indica en la salida del script.. De la cantidad de conexiones efectuadas, por lo menos la mitad de ellas debe ser balanceadas entre los dos nodos	De las conexiones hechas, el 50% están en la instancia #1 y las otras en la instancia #2	Pasa

Prueba Nro. 4

Objetivo: Probar la estabilidad de la base de datos al terminar (kill) procesos de servidores.

Tabla 3.22 Descripción de la Prueba Nro.4

Acción	Expectativa	Observación	Resultado
1. Desde el Pc Cliente conectarse a la base de datos a través de sqlplus: $ sqlplus HR/HR@RACDB 2. Verificar a cual instancia está conectado el usuario HR: SQL> select instance_name from v$instance; 3. Ejecutar una sentencia select por parte del usuario HR e inmediatamente matar (kill) un proceso relacionado a los demonios Oracle desde una terminal con el fin de bloquear los procesos de la base de datos cuya instancia este el usuario HR conectado HR: SQL> select * from test; root@rac1o2 #: ps –ef |grep smon |cut –c 10-15|xargs kill -9	La sesión SQL no debería verse afectadas y debe hacerse el failover hasta el nodo siguiente sin ninguna interrupción. Las salida del proceso de selección SELECT no debe ser afectado.		Pasa

B. PRUEBAS FAILOVER EN HARDWARE:

Prueba Nro. 1

Objetivo: Probar el failover o conmutación por fallo en la red privada o de interconexión

Tabla 3.23 Descripción de la Prueba Nro.1

Acción	Expectativa	Observación	Resultado
1. Identificar la tarjeta de red que tiene la dirección IP Privada en cada nodo 2. Hacer un ping de un nodo al otro nodo a través de la dirección privada. 3. Con el ping levantado, proceder a desconectar en la interfaz de red el cable de la red privada	El ping debe responder con los mensajes de retorno		Pasa

Prueba Nro. 2

Objetivo: Probar el failover o conmutación por fallo de la tarjeta de red asignada a la dirección IP Pública.

Tabla 3.24 Descripción de la Prueba Nro.2

Acción	Expectativa	Observación	Resultado
1. Identificar la tarjeta de red que tiene la dirección IP Privada en cada nodo 2. Hacer un ping desde el PC cliente hacia el nodo. 3. Sacar uno de los cables de interconexión	El ping debe ser capaz de mantenerse		Pasa

Con respecto a los resultados obtenidos de las pruebas, se concluye que la arquitectura RAC tiene una tolerancia inherente a fallas que se presente en algunos de sus nodos debido a que la aplicación corre en forma independiente en cada nodo físico; estas caídas son imperceptibles en su mayoría al usuario que esté conectado a la base de datos y efectuando consultas y perceptibles cuando el usuario está realizando procesos de escritura (intercesión y actualización de registros) como el bloqueo temporal de su sesión sin que esta se caiga o caída total de la sesión teniendo que restablecer una nueva conexión. Adicional a esto, tiene un rendimiento por el balanceo en las conexiones y una capacidad de ampliación sin tener que efectuar modificaciones a las aplicaciones.

Los niveles de disponibilidad actuales en los servicios web del portal y SIMCA son bastante altos, dado que en el periodo de observación no se presentaron eventos inesperados que afectaran dichos niveles, sin embargo estos pueden presentarse en cualquier momento y las arquitecturas para el despliegue de estos servicios no están preparadas para mitigar los efectos que traen los eventos de falla. La solución que se ha planteado en este capítulo, está en la capacidad de mejorarlos o cuando menos mantenerlos ya que la respuesta ante fallos es de tipo automático

minimizando por tanto los tiempos y demoras que una falla pueda ocasionar, siendo estos del orden de los segundos. Lo anterior puede verse en la Tabla 3.25 al efectuar un comparativo entre tablas 3.10 y 3.15 y las tablas 2.1 y 2.2.

Tabla 3.25 Comparativo de los tiempos de respuesta de elementos para los servicios Web y SIMCA.

Elemento	Tiempo de respuesta para la arquitectura actual.	Tiempo de respuesta para la solución tecnológica.
Caída de Base de Datos MYSQL	6,75 Minutos	3,33 Segundos
Caída del Nodo Balanceador de Portal	2,69 Minutos	3 Segundos
Caída del Nodo Balanceador de SIMCA	5,71 Minutos	3 Segundos

Además haciendo uso de la ecuación 1.1 es posible llegar a la tabla 3.26 donde se hace un comparativo entre la disponibilidad actual de los elementos y la disponibilidad de solución tecnológica implementada.

Tabla 3.26 Comparativo de la disponibilidad actual y lograda para cada elemento.

Elemento	Disponibilidad Actual	Disponibilidad para la solución tecnológica.
Caída de Base de Datos MYSQL	99,9945%	99,9978%
Caída del Nodo Balanceador de Portal	99,9980%	99,9992%
Caída del Nodo Balanceador de SIMCA	99,9904%	99,9961%

De lo anterior se puede concluir que la solución planteada aumenta la disponibilidad de los servicios Web y SIMCA al aumentar la disponibilidad de los elementos que la conforman.

3.6 IMPLEMENTACIÓN EN AMBIENTE REAL

Finalmente, para esta implementación fue necesario hacer unas modificaciones al diseño original ya que inicialmente se pensó permitir a cada uno de los proveedores del servicio de acceso de Internet pasara directamente sobre una de las IP virtuales creadas a fin de eliminar el balanceo por DNS, el esquema se muestra en la figura 3.7

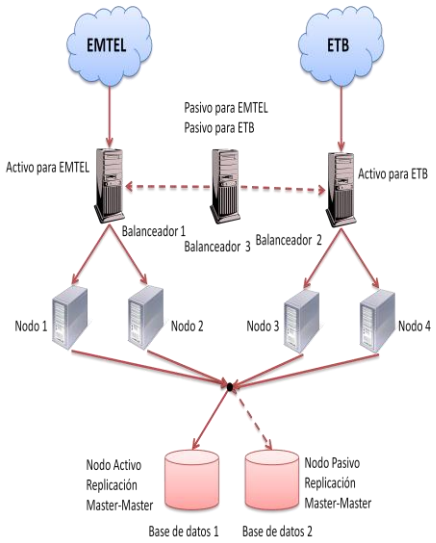

Figura 3.7 Solución Inicialmente Propuesta

Se determinó que la solución, al no balancear el tráfico desde la fuente, puede recargar una de las ramas de la solución, así que se optó por la descrita en la figura 3.9.

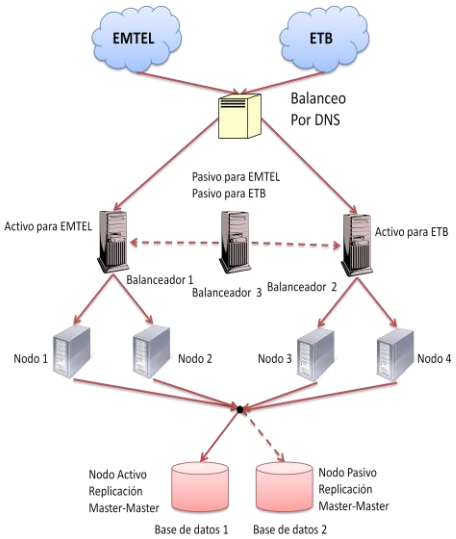

Figura 3.8 Solución Modificada

A pesar de que esta configuración continua usando el balanceador por DNS en la fuente, cabe destacar que en esta solución los problemas originalmente hallados ya no se encuentran, debido a que son mitigados directamente por la capa compuesta por HAProxy y Keepalived que implementa mecanismos de persistencia y el uso de una IP de tipo flotante, de tal forma que la caída o cambio de un balanceador es transparente para el DNS haciendo que la Disponibilidad recaiga como función propia de esta capa.

Otro cambio que fue necesario implementar sobre la configuración inicial se presentó dado el hecho de que se cuenta con dos proveedores de servicio de acceso a Internet, por lo cual se hizo

necesario el desarrollo de un script que le permita al balanceador de respaldo (Balanceador 3) cambiar su puerta de enlace predeterminada.

También es importante destacar, que para la replicación fue necesario hacer modificaciones en su configuración inicial ya que después de probarla y manipularla se demostró que debían hacerse especificaciones para su uso con el sistema de gestión de contenido (CMS) Drupal que es empleado en el portal.

3.7 METODOLOGÍA DE SOLUCIÓN

Para el desarrollo de la solución tecnológica se ha seguido una metodología que es una abstracción de la metodología en espiral, la cual, es un modelo de ciclo de vida desarrollado por Barry Boehm en 1986 y utilizado generalmente en la Ingeniería de software. Las actividades de este modelo son una espiral, donde cada bucle es una actividad.

Algunos principios básicos del modelo en espiral son:

- Decidir qué problema se quiere resolver antes de intentar resolverlo.
- Examinar las múltiples alternativas de acción y elegir las más convenientes.
- Evaluar lo que se ha hecho y lo que se ha aprendido durante el proceso de realización.

La elección de esta metodología obedeció al hecho de que permite una minimización de riesgos, ya que cada actividad está precedida por una menos compleja. Para su desarrollo se definieron las 4 actividades que se aprecian en la figura 3.10.

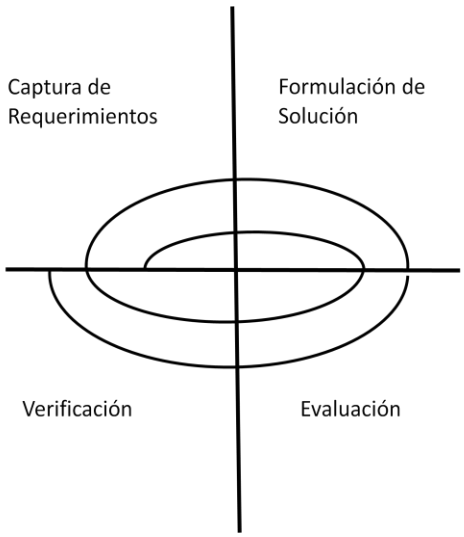

Figura 3.9 Metodología de la Solución

Para la primera Iteración es posible ver la captura de requerimientos que se desprendieron del diagnóstico realizado en el Capítulo 2, de esto se formuló una solución inicial que consistía en aplicar herramientas de alta disponibilidad y balanceo de carga para los puntos y falencias encontradas, es así como se procedió a una exploración tecnológica, consistente en buscar todas

aquellas soluciones existentes que tuvieran las característica funcionales requeridas, con lo que surgió el listado relacionado en la tabla 3.2, posteriormente se realizó la evaluación de dicho listado mediante la aplicación de la fase 1 de la metodología de evaluación OpenBRR obteniendo como resultado inicial la tabla 3.3, a la cual, se le efectuó un proceso de verificación y captura de requerimientos minimizando la cantidad de proyectos, los cuales se listan en la tabla 3.4; en la segunda iteración, se formulan las posibles soluciones tecnológicas con los proyectos resultantes de la primera iteración, como se observa en el apartado 3.3; los proyectos son evaluados mediante la aplicación de la fase 3 y 4 de la metodología de evaluación de calidad OpenBRR, se hace una verificación final en la que se valida la solución y se efectúa su montaje en una implementación en ambiente real. En la tercera iteración, surge una nueva lista de requerimientos para los que se hace una nueva formulación que incluye el diseño e implementación del script que se puede ver en el anexo E (Manuales de instalación y configuración), se verifica el comportamiento de la herramienta mediante pruebas de caída y verificación del estado, con lo cual se comprueba el estado satisfactorio de la solución tecnológica dada.

3.8 MEJORAS HARDWARE RECOMENDADAS

Adicionalmente a las soluciones presentadas para brindar Alta Disponibilidad y Balanceo de Carga, es necesario tener en cuenta las siguientes recomendaciones:

- Mejorar el etiquetado: este debe estar hecho al frente y atrás de los gabinetes y debe ser rotulado según normas de etiquetado, esto tiene como finalidad agilizar los procesos de cambio de cables físicos que puedan ser necesarios durante una crisis.
- Tener una documentación completa de todo el sistema de cableado.
- Mejorar redundancia de componentes (fuentes de alimentación, procesadores) en equipos críticos como enrutadores, switches para las redes LAN y SAN: A pesar de tenerse en algunos componentes del sistema, esto no se presenta en todos.
- Tener redundancia en el cableado del backbone LAN/SAN de conexión, ya sea en la fibra o en el par de cobre: La finalidad de esta recomendación es evitar puntos únicos de fallo.
- Separar el camino de ingreso de los proveedores en al menos 20 metros, sólo debe tener vías comunes en los espacios finales.
- Evitar que las conexiones redundantes estén en las mismas fundas del cable.
- Poner redundancia en servidores de monitoreo y control.
- Proporcionar al menos redundancia N+1 para el sistema, incluyendo el generador y sistema de UPS: Esto tiene como finalidad dar autonomía a las maquinas en caso de corte eléctrico.
- Buscar la forma que el edificio tenga al menos dos alimentadores de diferentes subestaciones eléctricas.
- Instalar un supresor de picos de tensión transitorios (TVSS) en todos los niveles del sistema de distribución que alimenta a las cargas electrónicas críticas.
- Implementar sistemas de monitoreo ambiental y de alimentación eléctrica a todos los equipos principales, como los dispositivos de distribución principal, sistemas generadores, sistemas UPS, interruptores automáticos de transferencia estática (ASTS), unidades de distribución de

energía, interruptores de transferencia automática, centros de control de motores, sistemas de supresión de aumento transitorio de voltaje, y sistemas mecánicos.

- Evitar que un circuito eléctrico sirva a más de un bastidor.
- Implementar sistemas y políticas para que en caso de falla durante un traspaso de alimentación eléctrica, se evite que las cargas electrónicas críticas sufran interrupción en la alimentación.
- Implementar un sistema de monitoreo de baterías para todo el sistema crítico.
- Oracle Corporation recomienda el uso de equipos certificados para la implementación de su solución Oracle RAC 11g, en la tabla 3.25, se muestra los equipos y la inversión necesaria para adquirirlos.

Tabla 3.27 Plan de Hardware para Oracle RAC 11g R2

PLAN DE EQUIPOS ORACLE			
Tipo de equipo	**Nombre de Configuración**	**Equipo**	**Costo a la Fecha**
Nodo 1	rac1	PowerEdge R310	$3.331.586
Nodo 1	rac2	PowerEdge R310	$3.331.586
Almacenamiento	storage	MD1000	$10.516.695
COSTO TOTAL			**$ 17.179.867**

Como conclusión de este capítulo se puede destacar que a pesar de que fue necesaria la implementación de cambios en la solución originalmente planteada los mismos no cambian la esencia, constituyéndose en una mejora frente a lo actualmente implementado ya que elimina los puntos de falla mediante la automatización de respuestas frente fallos.

CAPITULO 4: CONCLUSIONES Y TRABAJO FUTURO

4.1 CONCLUSIONES

1. La solución tecnológica de Alta Disponibilidad y Balanceo de Carga diseñada con proyectos de uso libre cumple tanto con los aspectos económicos como técnicos constituyéndose en una mejora frente a lo que actualmente está implementado según los resultados obtenidos en las pruebas realizadas en las implantaciones en ambientes virtual y real.
2. La solución tecnológica diseñada puede ser implementada en equipos de bajo costo convirtiéndose en una alternativa para pequeñas y medianas empresas que no cuentan con recursos para grandes inversiones.
3. La elección e implementación la metodología en espiral permite refinar el producto final al hacer una verificación de los resultados obtenidos, minimizando también los riesgos ya que cada actividad está precedida por una menos compleja.
4. Con la eliminación de puntos únicos de fallo (SPOF), el suministro de redundancia en cada capa, duplicación de equipos, sincronización de datos y la configuración de procesos de automatización de tolerancia ante fallos se logran mejoras a la disponibilidad.
5. Las comunidades de desarrollo de Software Libre tiene un alto dinamismo tanto en el continuo mejoramiento de proyectos existentes como en la creación de nuevas iniciativas generando muchas alternativas por lo que es importante emplear un modelo de evaluación de Calidad de Software libre como OpenBRR que permite examinar de forma rápida y sistemática mediante un conjunto de criterios para evaluarlas y seleccionar una o varias de ellas para diseñar e implantar una solución tecnológica.

4.1 TRABAJO FUTURO

- ✓ Realizar un monitoreo constante de las diferentes metodologías de evaluación de Calidad de Software Libre.
- ✓ Hacer un paralelo entre sistemas de alta disponibilidad reales y de virtualización, revisando con ello sus características y velocidades de respuesta.
- ✓ Hacer un estudio costo beneficio sobre la posibilidad de migración hacia tecnología Cloud Computing.
- ✓ Realizar una revisión de las nuevas iniciativas de software Libre, algunas prometedoras como: Proxmox VE version 2.X o superior, Percona XtraDB Cluster para base de datos MySQL, que rápidamente van alcanzando cierta madurez a fin de constar en qué grado pueden mejorar la Alta Disponibilidad y/o Balanceo de Carga.
- ✓ Se evidencia que, a pesar de que la red de datos cuenta con la herramienta Nagios para monitoreo y generación de alarmas, no es suficientemente eficaz para el reporte de incidencias que se presentan, se sugiere como trabajo futuro, el diseño e implementación de un sistema de monitoreo robusto y sofisticado que pueda tomar ciertas decisiones ya programadas ante eventos y alarmas activadas y/o reporte inmediatamente a los

administradores; dicho monitoreo no solo debe limitarse a los servidores y equipos de comunicación sino también elementos del sistema como lo es el aire acondicionado, el sistema anti-incendios, el sistema de alimentación eléctrico, los sistemas de respaldo de energía, etcétera.

BIBLIOGRAFÍA

[1] J. Vidal y S. Cubero, "Alta disponibilidad gracias a las tecnologías de virtualización y redes", *rediris.es*, [En línea]. Disponible en: http://www.rediris.es/difusion/publicaciones/boletin/82-83/ponencia1.4B.pdf. [Accedido: Abr. 2, 2011].

[2] Universidad del Cauca, "SIMCA home page", unicauca.edu.co, [En línea]. Disponible en: https://simca.unicauca.edu.co/simca/. [Accedido: Ene. 10, 2012].

[3] S. M. Pantoja y J. J. Imbachí, "Propuesta de Solución de Alta Disponibilidad de los servicios Críticos de Centro de Datos de la Universidad del Cauca: Anexo C", Trabajo de grado, Universidad del Cauca, Popayán, Colombia, 2010.

[4] C. Lincovil y I. Gutiérrez, "Optimización Económica de la Disponibilidad", Departamento de Ingeniería Eléctrica de la Escuela de Ingeniería de la Pontificia Universidad Católica de Chile. [En línea]. Disponible en: http://web.ing.puc.cl/~power/alumno06/OED/archivos/Optimizacion_Economica_de_la_Disponibilidad.pdf. [Accedido: Abr. 2, 2011].

[5] Atos Origin, "Method for Qualification and Selection of Open Source software (QSOS) version 1.6", *qsos.org*, Oct. 23, 2006. [En línea]. Disponible en: http://www.qsos.org/?page_id=3. [Accedido: Abr. 18, 2011].

[6] B. Russo, M. Scotto, A. Sillitti, y G. Succi, *Agile Technologies in open Source Development*. Hershey: IGI Global, 2010, p. 302.

[7] A. Valdivia, E. Reinoso, y C. Diaz, "¿Qué es un Cluster?", *es.scribd.com*, Oct. 16, 2008. [En línea]. Disponible en: http://es.scribd.com/doc/6858172/QUE-ES-UN-CLUSTER. [Accedido: Abr. 1, 2011].

[8] I. Bernal, D. Mejía y D. Fernández, "Computación de Alto Rendimiento con *Clusters* de PCs", I Congreso Internacional de Ciencia y Tecnología, Oct. 2005, Quito, Ecuador. [En línea]. Disponible en: http://clusterfie.epn.edu.ec/clusters/Publicaciones/Download/ComputacionAltoRendimiento.pdf. [Accedido: Ene. 20, 2012].

[9] A. G. Valdez y G. Campos, "Creación de un Cluster de Linux utilizando Knoppix", Revista Digital Sociedad de la Información, no. 13, Abr. 2008. [En línea] Disponible en: http://sociedadelainformacion.com/13/cluster.pdf. [Accedido: Abr. 2, 2011].

[10] B. Stansberry, G. Zamarreno y P. Ferraro, "Clustering Concepts", *JBoss Application Server 5 Clustering Guide*, S. Kittoli, 2009. [En línea]. Disponible en: http://docs.jboss.org/jbossclustering/cluster_guide/5.1/html/cluster.concepts.chapt.html. [Accedido: Abr. 2, 2011]

[11] F. J. Perozo, "Cluster Mangosta: Implementación y Evaluación", FARAUTE de Ciencias y Tecnología, vol. 1, no. 1, Jul. 2006. [En línea]. Disponible en: http://servicio.bc.uc.edu.ve/facyt/v1n1/1-1-2.pdf. [Accedido: Abr. 8, 2011]

[12] F. M. García, "Qué es un cluster? Oracle RAC Notes", Blog Oracle RAC Notes, Feb. 20, 2008. [En línea]. Disponible en: http://oracleracnotes.wordpress.com/2008/02/20/que-es-un-cluster/. [Accedido: Jun. 25, 2011].

[13] P. Clavijo, "Introducción a los Cluster", lintips.com, Ago. 2, 2010. [En línea]. Disponible en: http://www.lintips.com/?q=node/117. [Accedido: Abr. 2, 2011].

[14] C. Sánchez, "Clusters", notas de clase para Cátedra Gestión de Conocimiento, Escuela de Economía y Negocios, Facultad de Ciencias Económicas y Administrativas, Universidad Central. [En línea]. Disponible en: http://www.slideshare.net/cduranmardones/trabajo-clusters. [Accedido: Abr. 2, 2011].

[15] A. Borrero, "Firewall en cluster de alta disponibilidad", Proyecto Integrado, IES Gonzalo Nazareno, Sevilla, España, 2011. [En línea]. Disponible en: http://informatica.gonzalonazareno.org/proyectos-ASI/2010-11/proyecto_integrado_arturo_borrero_pdf.pdf. [Accedido: Ene. 10, 2012].

[16] Microsoft, "Descripción de la disponibilidad, la confiabilidad y la escalabilidad", *technet.microsoft.com*. [En línea]. Disponible en: http://technet.microsoft.com/es-es/library/aa996704(EXCHG.65).aspx. [Accedido: Abr. 1, 2011].

[17] G. F. Castillo, "Sistemas de bases de datos de Alta Disponibilidad", Tesis de Grado, Universidad de San Carlos de Guatemala, Ciudad de Guatemala, Guatemala, 2006.

[18] S. M. Pantoja y J. J. Imbachí, "Propuesta de Solución de Alta Disponibilidad de los servicios Críticos de Centro de Datos de la Universidad del Cauca", Trabajo de grado, Universidad del Cauca, Popayán, Colombia, 2010.

[19] E. Marcus y H. Stern, *Blueprints for High Availability*, 2nd ed. Indianapolis: Wiley Publishing, Inc, 2003.

[20] Osiatis, "Fundamentos de la Gestión de servicios de TI. Gestión de la disponibilidad, métodos y técnicas", *itil.osiatis.es*. [En línea]. Disponible en: http://itil.osiatis.es/Curso_ITIL/. [Accedido: Mar. 28, 2011].

[21] S. Cujano, "ANÁLISIS E IMPLEMENTACIÓN DE ALTA DISPONIBILIDAD MEDIANTE CLUSTERING EN SISTEMAS DE CALL CENTER BASADOS EN VoIP", Tesis de Grado, Escuela Superior Politécnica de Chimborazo, Facultad De Informática Y Electrónica, Escuela De Ingeniería Electrónica En Telecomunicaciones Y Redes, Riobamba, Ecuador, 2011.

[22] E. Ciurana, "Scalability and high availability", *dzone.com*. [En línea]. Disponible en: http://library.dzone.com/sites/all/files/refcardz/rc043-010d-scalability_3.pdf. [Accedido: Mar. 28, 2011].

[23] T. Bourke, *Server Load Balancing*. USA: O'Reilly & Associates, Inc., 2001.

[24] F. J. Luna, R. Tristán, y J. J. Martínez, "Aplicando un Algoritmo Genético para Balancear Carga Dinámicamente en Ambientes Distribuidos Orientados a Objetos (CORBA)", *Conciencia Tecnológica*, no. 23, 2003. [En línea]. Disponible en: http://redalyc.uaemex.mx/pdf/944/94402306.pdf. [Accedido: Abr. 27, 2011].

[25] C. Kopparapu, *Load Balancing Servers, Firewalls and caches*, USA: John Wiley & Son, Inc., 2002.

[26] J. Balasubramanian, D. C. Schmidt, L. Dowdy, y O. Othman, "Evaluating the Performance of Middleware Load Balancing Strategies", Proc. of the 24th IEEE Intl. Conference on Distributed Computing Systems, 2004, Nashville, TN, USA, pp. 135-146. [En línea]. Disponible en: http://www1.cse.wustl.edu/~schmidt/PDF/ICDCS_2004.pdf. [Accedido: Mar. 28, 2011].

[27] "Server Load Balancing", *radharc.com*. [En línea]. Disponible en: http://www.radharc.com/whitepapers/Server_Load_Balancing.pdf. [Accedido: Abr. 2, 2011].

[28] J. Pappalexis, "Why Load Balance? An overview for Users of Web Security Products", Cupertino, CA, USA: TrendEdge Publications, 2008.

[29] "Qué es balanceo de carga en una red?", *ordenadores-y-portatiles.com*, [En línea]. Disponible en: http://www.ordenadores-y-portatiles.com/balanceo-de-carga.html. [Accedido: Jun. 25, 2011].

[30] "What is Global Server Load Balancing and how it works?", *eukhost.com*. [En línea]. Disponible en: http://www.eukhost.com/web-hosting/kb/global-server-load-balancing/. [Accedido: Jun. 24, 2011].

[31] Pamcho, "Wackamole: Alta Disponibilidad con Balanceo de Carga", *11870.com*, Abr. 29, 2009. [En línea]. Disponible en: http://11870.com/2e5e/2009/04/29/wackamole-alta-disponibilidad-con-balanceo-de-carga/. [Accedido: Jun. 27, 2011].

[32] R. Sanjay, "Evaluating Server Load Balancing Solutions", *cmg.org*. [En línea]. Disponible en: http://www.cmg.org/proceedings/2003/3080.pdf. [Accedido: Nov. 2010]

[33] Neurowork, "WhyFLOSS FLOSS evangelization - ¿ Por qué FLOSS ?", *whyfloss.com*. [En línea]. Disponible en: http://www.whyfloss.com/. [Accedido: Jun. 20, 2012].

[34] Free Software Foundation, "The Free Software Definition", *gnu.org*. [En línea]. Disponible en: http://www.gnu.org/philosophy/free-sw.html. [Accedido: Abr. 30, 2011].

[35] Open Source Initiative, "The Open Source Definition (Annotated)", *opensource.org*. [En línea]. Disponible en: http://www.opensource.org/osd.html. [Accedido: Abr. 30, 2011].

[36] J. M. Gonzalez-Barahona, *Good practices for R&D projects producing FLOSS (Request for comments)*, GSyC/LibreSoft, Universidad Rey Juan Carlos, Madrid, España, 2010.

[37] Escuela PNUD, "¿Qué es el software libre?", *escuelapnud.org*. [En línea]. Disponible en: http://www.escuelapnud.org/files/pub_pages/about_us/attachments/software_libre_v2.pdf. [Accedido: Jul. 9, 2011].

[38] J. Gonzales-Barahona, J. Seoane, y G. Robles, *Introducción al software Libre*, Universitat Oberta de Catalunya. [En línea] Disponible en: http://curso-sobre.berlios.de/introsobre/1.0/libre.pdf [Accedido: Jun. 22, 2011]

[39] K. Stol y M. Ali Babar, "A comparison framework for open source software evaluation methods", in *Open Source Software: New Horizons 6th International IFIP WG 2.13 Conference on Open Source Systems*, OSS 2010, 1st ed., vol. 319. Notre Dame, IN, USA: Springer-Verlag, 2010, pp. 389-394.

[40] B. Russo, M. Scotto, A. Sillitti, y G. Succi, Agile Technologies in *open Source Development*.: Hershey: IGI Global, 2010, p. 302.

[41] O. Fernández,, D. García y A. Beltrán, "Un enfoque actual sobre la calidad del software", in *ACIMED*, vol. 3, no. 3, Dic. 1995. [En línea]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1024-94351995000300005&lng=es&nrm=iso. [Accedido: Abr. 15, 2011].

[42] D. A. Wheeler, "How to Evaluate Open Source Software / Free Software (OSS/FS) Programs", *dwheeler.com*. [En línea] Disponible en: http://www.dwheeler.com/oss_fs_eval.html. [Accedido: Abr. 30, 2011].

[43] C. Montoya Becerra, "Evaluación de la Calidad de Proyectos de Software Libre", *soflibreocmb.blogspot.com*, Abr. 30, 2009. [En línea]. Disponible en: http://soflibreocmb.blogspot.com/2009/04/evaluacion-de-la-calidad-de-proyectos.html. [Accedido: Abr. 23, 2011].

[44] F. Duijnhouwer y C. Widdows, "Open Source Maturity Model", Capgemini Expert Letter, Ago. 2003.

[45] T. Koponen y V. Hotti, "Evaluation framework for open source software", in *Proc. Software Engineering and Practice (SERP)*, 2004, Las Vegas, Nevada, USA.

[46] G. Polančič y R.V. Horvat, "A Model for Comparative Assessment Of Open Source Products", in *Proc. The 8th World Multi-Conference on Systemics, Cybernetics and Informatics*, 2004, Orlando, USA.

[47] G. Polančič, R.V. Horvat, y T. Rozman, "Comparative assessment of open source software using easy accessible data", in *Proc. 26th International Conference on Information Technology Interfaces*, Jun. 7-10, 2004, Cavtat, Croatia, pp. 673 - 678.

[48] B. Golden*, Succeeding with Open Source*. Addison-Wesley, 2004.

[49] D. Woods y G. Guliani, *Open Source for the Enterprise: Managing Risks Reaping Rewards.* O'Reilly Media, Inc., 2005.

[50] A. I. Wasserman, M. Pal, y C. Chan, "Business Readiness Rating Project", "BRR Whitepaper 2005 RFC 1". [En línea] Disponible en: http://pascal.case.unibz.it/retrieve/1095/BRR_whitepaper_2005RFC1.pdf [Accedido: Abr. 12, 2011]

[51] A. I. Wasserman, M. Pal, y C. Chan, "The Business Readiness Rating Model: an Evaluation Framework for Open Source", in *Proc. Of Workshop on Evaluation Techniques for Open Source Software, 2nd Int'l. Conf. on Open Source Systems*, 2006, Como, Italia.

[52] Atos, Origin. "Method for Qualification and Selection of Open Source software (QSOS) version 1.6", *qsos.org*, Oct. 23, 2006. [En línea]. Disponible en: http://www.qsos.org/?page_id=3. [Accedido: Abr. 18, 2011].

[53] D. Cruz, T. Wieland, y A. Ziegler, "Evaluation criteria for free/open source software products based on project analysis", *Software Process: Improvement and Practice*, no. 2, Nov. 2006.

[54] W.J. Sung, J. H. Kim, y S. Y. Rhew, "A Quality Model for Open Source Software Selection", in *Proc. Sixth International Conference on Advanced Language Processing and Web Information Technology*, 2007, Luoyang, Henan, China, pp. 515-519.

[55] Y. M. Lee, J. B. Kim, I. W. Choi, y S. Y. Rhew, "A Study on Selection Process of Open Source Software", in *Proc. Sixth International Conference on Advanced Language Processing and Web Information Technology (ALPIT)*, 2007, Luoyang, Henan, China.

[56] M. Cabano, C. Monti, y G. Piancastelli, "Context-Dependent Evaluation Methodology for Open Source Software", in *Proc. Third IFIP WG 2.13 International Conference on Open Source Systems (OSS 2007)*, 2007, Limerick, Ireland, pp. 301-306.

[57] M. Cabano, C. Monti, y G. Piancastelli, "Valutazione del grado di maturità di soluzioni software Open Source", *oitos.it*, 2006. [En línea] Disponible en: http://www.oitos.it/opencms/opencms/oitos/Valutazione_di_prodotti/Modello1.2.pdf [Accedido: Abr. 12, 2011]

[58] C. A. Ardagna, E. Damiani, y F. Frati, "FOCSE: An OWA-based Evaluation Framework for OS Adoption in Critical Environments", in *Proc. Third IFIP WG 2.13 International Conference on Open Source Systems*, 2007, Limerick, Ireland, pp. 3-16.

[59] L. Lavazza, "Beyond Total Cost of Ownership: Applying Balanced Scorecards to Open-Source Software", in *Proc. International Conference on Software Engineering Advances (ICSEA) Cap Esterel*, 2007, French Riviera, France, p. 74.

[60] D. Taibi, L. Lavazza, y S. Morasca, "OpenBQR: a framework for the assessment of OSS", in *Proc. Third IFIP WG 2.13 International Conference on Open Source Systems (OSS 2007)*, 2007, Limerick, Ireland, pp. 173-186.

[61] R. Carbon, M. Ciolkowski, J. Heidrich, I. John, y D. Muthig, "Evaluating Open Source Software through Prototyping", in *Handbook of Research on Open Source Software: Technological, Economic, and Social Perspectives' (Information Science Reference, 2007)*, St. Amant y B. Still, Eds. Ed., 2007, pp. 269-281.

[62] M. Ciolkowski y M. Soto, "Towards a Comprehensive Approach for Assessing Open Source Projects", in *Software Process and Product Measurement*.: Springer-Verlag, 2008.

[63] I. Samoladas, G. Gousios, D. Spinellis, y I. Stamelos, "The SQO-OSS Quality Model: Measurement Based Open Source Software Evaluation", in *Proc. Fourth IFIP WG 2.13 International Conference on Open Source Systems (OSS 2008)*, 2008, Milano, Italy.

[64] A. Majchrowski y J. Deprez, "An operational approach for selecting open source components in a software development project", in *Proc. 15th European Conference, Software Process Improvement (EuroSPI)*, Sep. 3-5, 2008, Dublin, Ireland.

[65] V. del Bianco, L. Lavazza, S. Morasca, y D. Taibi, "Quality of Open Source Software: The QualiPSo Trustworthiness Model", in *Proc. Fifth IFIP WG 2.13 International Conference on Open Source Systems (OSS 2009)*, Jun. 3-6, 2009, Skövde, Suecia.

[66] V. del Bianco, L. Lavazza, S. Morasca, y D. Taibi, "The observed characteristics and relevant factors used for assessing the trustworthiness of OSS products and artefacts", Skövde, Suecia, Technical Report A5.D1.5.3, 2008.

[67] E. Petrinja, R. Nambakam, y A. Sillitti, "Introducing the OpenSource Maturity Model", in *Emerging Trends in Free/Libre/Open Source Software Research and Development*, 2009. FLOSS '09. ICSE Workshop on, 2009, Vancouver, Canadá, pp. 37 - 41.

[68] K. Haaland, A. Groven, R. Glott, y A. Tannenberg, "Free/Libre Open Source Quality Models-a comparison between two approaches", in *4th FLOSS International Workshop on Free/Libre Open Source Software*, Jul. 2010, Jena, Thuringia, Germany.

[69] B. Russo, M. Scotto, A. Sillitti, y G. Succi, *Agile Technologies in open Source Development.* Hershey: IGI Global, 2010, p. 304.

[70] National Resource Centre for FOSS, "Open Source Maturity Model for a Department/Enterprise/Institution", *nrcfoss.au-kbc.org.in*, [En línea]. Disponible en: http://www.nrcfoss.au-kbc.org.in/maturity/. [Accedido: Jun. 21, 2011]

[71] CapGemini, "Open Source Maturity Model: een selectie voor een open wereld", *nl.capgemini.com*, Sep. 2008. [En línea]. Disponible en: http://www.nl.capgemini.com/expertise/publicaties/open-source-maturity-model-een-selectie-voor-een-open-wereld/. [Accedido: Jun. 21, 2011]

[72] J. A. J. Wilson, "Open Source Maturity Model", *oss-watch.ac.uk*, Ene. 2011. [En línea] Disponible en: http://www.oss-watch.ac.uk/resources/osmm.xml?style=text [Accedido: Jun. 21, 2011]

[73] A. Das y A. I. Wasserman, "Using FLOSSmole Data in Determining Business Readiness Ratings", in *Workshop on Public Data about Software Development - WoPDaSD 2007*, 2007, Limerick, Irlanda.

[74] European Commission, "CORDIS FP6: What is FP6", *cordis.europa.eu*. [En línea]. Disponible en: http://cordis.europa.eu/fp6/whatisfp6.htm. [Accedido: Jun. 20, 2011].

[75] Centre d'Excellence en Technologies de l'Information et de la Communication - CETIC, "QualOSS QUALity in Open Source Software", *cetic.be*. [En línea]. Disponible en: http://www.cetic.be/article500.html. [Accedido: Jun. 20, 2011].

[76] J. Deprez, "QualOSS Final Activity Report", *qualoss.org*, Ene. 2010. [En línea] Disponible en: http://www.qualoss.org/deliverables/QualOSS_Final_Activity_Report_Submitted.pdf. [Accedido: Jun. 21, 2011].

[76] J. Tabilo, "GMQ - Gestión de Software", *fing.edu.uy*. [En línea]. Disponible en: http://www.fing.edu.uy/inco/cursos/gestsoft/Presentaciones/GQM%20-%20G9/GQM.doc. [Accedido: Jul. 2011, 18].

[78] M. Wittmann y R. Nambakam, "Deliverable A6.D1.6.3 CMM-like model for OSS", *qualipso.org*, Nov. 2008. [En línea]. Disponible en: http://www.qualipso.org/sites/default/files/A6.D1.6.3CMM-LIKEMODELFOROSS.pdf. [Accedido: Jun. 21, 2011].

[79] E. Petrinja, A. Sillitti, y G. Succi, "Overview on Trust in large FLOSS Communities", in *IFIP International Federation for Information Processing*, Sep. 2008, Milan, Italy, pp. 47-56.

[80] G. Gousios et al., "Software Quality Assessment of Open Source", *istlab.dmst.aueb.gr*. [En línea]. Disponible en: http://istlab.dmst.aueb.gr/~george/pubs/2007-PCI-GKSLVS/paper.pdf. [Accedido: Jul. 15, 2011].

[81] R. V Solingen, "The goal/question/metric", in *Encyclopedia of Software Engineering 2*, 2002, pp. 578–583.

[82] V. R. Basili, "Software Modeling and Measurement: The Goal/Question/Metric Paradigm", Computer Science Technical Report Series NR: CSTR2956/ NR: UMIACSTR9296, 1992.

[83] D. Spinellis et al., "Evaluating the quality of open source software", in *SQM 2008: Second International Workshop on Software Quality and Maintainability—12th European Conference on Software Maintenance and Reengineering (CSMR 2008) satellite event*, Mar. 2008, Atenas, Grecia, pp. 5 - 28.

[84] K. Stol y M. Ali Babar, "A comparison framework for open source software evaluation methods", in *Open Source Software: New Horizons 6th International IFIP WG 2.13 Conference on Open Source* Systems, OSS, 2010, 1st ed., vol. 319. Notre Dame, IN, USA: Springer-Verlag, 2010, pp. 389-394. [En línea]. Disponible en: http://hdl.handle.net/10344/748. [Accedido: Jul. 15, 2011].

[85] K. Stol, *Supporting Product Development with Software from the Bazaar*. PhD [Dissertation]. Limerick, Ireland: University of Limerick, 2011. [En línea]. Disponible en: http://staff.lero.ie/stol/files/2011/12/stol_phd_2011.pdf. [Accedido: Jul. 15, 2011].

[86] E. Petrinja, A. Sillitti, y J. Succi, "Comparing OpenBRR, QSOS, and OMM Assessment Models", in *6th International Conference on Open Source Systems (OSS2010)*, 2010, Notre Dame, Indiana, pp. 224-238.

[87] J. Deprez y S. Alexandre, "Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR and QSOS", *qualoss.org*,. [En línea]. Disponible en: http://www.qualoss.org/dissemination/DEPREZ_CompareFlOSSAssessMethodo-Camera-02.pdf. [Accedido: May. 16, 2011].

[88] A. Toledo, J. V. Cuellar, J. R. Romero, "Análisis de modelos de evaluación de calidad de software libre", en *Proc. II Encuentro Internacional y VI Nacional de Investigación en Ingeniería de Sistemas e Informática - EIISI 2011*, Tunja, CO, 2011, pp. 95–106.

[89] ZonaDiegum, "Sistemas de Misión Crítica (I): Acuerdos de Nivel de Servicio (SLAs)", [En linea]. Disponible en: http://diegumzone.wordpress.com/2007/01/13/sistemas-de-mision-critica-i-acuerdos-de-nivel-de-servicio-slas/ [Accedido: Ene. 10, 2012]

[90] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison-Wesley 3th edition, p.p. 33

[91] E. Villar, "Virtualización de servidores de telefonía IP en GNU/Linux," Trabajo de Titulación, Universidad de Almería, Almería, España, 2010. [En línea]. Disponible en: http://www.adminso.es/images/d/dc/PFC_eugenio.pdf. [Accedido: Ene. 10, 2012]

[92] T. Burke, "High Availability Cluster Checklist," in Linux Journal, no. 80, Dic. 2000. [En línea]. Disponible en: http://www.linuxjournal.com/article/4344?page=0,0. [Accedido: Oct. 10, 2011].

[93] Drupal™, "System requirements," *drupal.org*. [En línea]. Disponible en: http://drupal.org/node/270/&usg=A. [Accedido: Oct. 2, 2011]

[94] Drupal Mail List, "Does Drupal support MySQL Cluster?," *lists.drupal.org*. [En línea]. Disponible en: http://lists.drupal.org/pipermail/support/2009-July/012599.html. [Accedido: Oct. 8, 2011].

[95] Oracle®, "6.4. Cómo montar la replicación," *dev.mysql.com*. [En línea]. Disponible en: http://dev.mysql.com/doc/refman/5.0/es/replication-howto.html. [Accedido: Oct. 10, 2011].

[96] Oracle®, "B.10.3: What is the difference between using MySQL Cluster vs using MySQL Replication?," *dev.mysql.com.* [En línea]. Disponible en: http://dev.mysql.com/doc/refman/5.0/en/faqs-mysql-cluster.html#qandaitem-B-10-1-3. [Accedido: Oct. 10, 2011].

[97] J. J. Gutiérrez, "Generación de pruebas de sistema a partir de la especificación funcional," Informe de Investigación, Universidad de Sevilla, Sevilla, España, 2005. [En línea]. Disponible en: http://www.lsi.us.es/~javierj/investigacion_ficheros/NDEAv25.pdf. [Accedido: Dic. 27 de 2011].

[98] I. Fernández, J. Iñiguez y F. Mitchell, "Diseño de un Sistema de Mantenimiento para la casa de Fuerza del Hospital Regional Vicente Corral Moscoso," *dspace.ups.edu.ec*, Feb. 11 de 2011. [En línea]. Disponible en: http://dspace.ups.edu.ec/bitstream/123456789/829/1/Abstract.pdf. [Accedido: Ago. 20, 2011].

[99] M. A. Calderón, C. F. Corral, "Desarrollo de una Herramienta para la creación y administración de Clústeres Computacionales para Simulaciones FDTD (Finite-Difference Time-Domain) con el Paquete Meep, sobre el servicio Elastic Compute Cloud (Ec2) de Los Amazon Web Services (Aws)," Informe de materia de Graduación, Escuela Superior Politécnica del Litoral, Guayaquil, Ecuador, 2010.

[100] B. Merino, "Análisis de Tráfico con Wireshark," *cert.inteco.es*. [En línea]. Disponible en: http://cert.inteco.es/extfrontinteco/img/File/intecocert/EstudiosInformes/cert_inf_seguridad_analisis_trafico_wireshark.pdf. [Accedido: Ene. 13, 2012].