UACM

Universidad Autónoma de la Ciudad de México

NADA HUMANO ME ES AJENO

COLEGIO DE CIENCIA Y TECNOLOGÍA LICENCIATURA EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS Y DE TELECOMUNICACIONES

Instalación y configuración de un balanceador de carga para el manejo de varios servidores web

T E S I S

QUE PARA OPTAR POR EL TÍTULO DE

LICENCIADOS EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS Y DE TELECOMUNICACIONES

P R E S E N T A

MIRIAM VIDAL GRANADOS

DAVID EMMANUEL CONTRERAS BARRAGÁN

D I R E C T O R

M. EN I. OSCAR RENE VALDEZ CASILLAS

Ciudad de México, marzo de 2025.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©


UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO- CASA LIBERTAD -
- CCyT -
INGENIERÍA EN SISTEMAS ELECTRÓNICOS Y DE TELECOMUNICACIONES
UACM

JURADO ASIGNADO

Presidente: Dra. Tejinder Kaur

Secretario: Lic. David Velázquez Suaréz

Vocal: Mtr. En Ing. Oscar René Valdez Casillas

Lugar donde se realizó examen de titulación: Universidad Autónoma de la Ciudad de México.

Plantel Casa Libertad

Modalidad: Presencial

RESUMEN

El presente trabajo aborda el análisis y evaluación del desempeño de un balanceador de carga en un entorno de servidores distribuidos. Con el creciente aumento del tráfico en aplicaciones y sitios web, la implementación de balanceadores se ha convertido en una estrategia para optimizar la distribución de carga, mejorar la disponibilidad del servicio y evitar la saturación de los servidores.

A lo largo de este estudio, se exploraron herramientas como **Netdata** y **Siege**, que permitieron monitorear, evaluar y examinar el rendimiento del sistema desde distintas perspectivas. Se llevaron a cabo pruebas con diferentes configuraciones y cargas de tráfico en la red, generadas en un ambiente controlado, considerando métricas como CPU, disponibilidad y transacciones exitosas y fallidas.

El documento se estructura en capítulos que detallan desde los fundamentos teóricos hasta los resultados experimentales, proporcionando una visión integral sobre la implementación, configuración y análisis del balanceador de carga.

En el capítulo 1 se presenta las bases en cuestiones teóricas cómo redes de computadoras, Suite de protocolos, servidores web, balanceadores de carga, además, planteamiento del problema, justificación y metodología.

En el capítulo 2 se muestra la configuración de un ambiente local controlado, se puede notar que es indispensable el conocimiento de las redes de computadoras que se abordó en el capítulo anterior. El desarrollo de la red LAN fue necesaria para la comunicación entre todos los dispositivos, ya que sin esta no habría interacción entre ellos.

En el capítulo 3 se establece la línea base, se describen las métricas que miden el rendimiento, así mismo se explica y se eligen los softwares que ayudarán a demostrar detalladamente el desempeño del balanceador de carga.

En el capítulo 4 se analizan, comparan y presentan los resultados obtenidos a lo largo de los capítulos anteriores.

En el capítulo 5 se presentan las conclusiones a partir de los resultados obtenidos en los capítulos anteriores, evaluando si se alcanzan los objetivos establecidos. Además, se plantea el trabajo futuro, identificando nuevas áreas de desarrollo que pueden enfocarse en aspectos más específicos según el proyecto a realizar.

AGRADECIMIENTOS

Miriam Vidal Granados

Hoy que finalizo una etapa más en mi vida quiero ofrecer mis más sinceros agradecimientos.

Primeramente, a Dios mi padre celestial, mi amado Cristo Jesús y su hermoso Espíritu Santo Por estar siempre conmigo y permitirme llevarlos en mi corazón, por hacer lo imposible posible, pues teniendo todo en contra, Él fue mi fortaleza, quien me sostuvo en mi debilidad, llenando mi vida de salud y sabiduría para terminar esta etapa en mi vida.

A mi amada familia

Mi mamita hermosa Gloria, hermanitos Adán y José. Por todos los sacrificios y desvelos que pasaron por mí. Por todo su amor, sus oraciones, apoyo, consejos y confianza que me brindaron siempre, porque siempre han estado junto a mi cuando más los he necesitado. Son mi más grande admiración, motivo y ejemplo a seguir en todos los sentidos. A mi pequeño angelito Isaac, eres ese motorcito que cada día alegra mi corazón y lo anima a seguir adelante, te amo.

A mis profesores

Macías, por el tiempo que me permitió trabajar en su cubículo y el apoyo que me brindó, a los profesores Hong, Song, Manuel, Aranda, Ocampo, Iván, Tejinder, David Estrada y Velázquez por todas sus enseñanzas, esfuerzo y dedicación que pusieron en mi formación académica, han sido ejemplo de superación en mi vida, pues me brindaron herramientas para seguir adelante dentro y fuera de la UACM. Un agradecimiento especial, a mi Director de Tesis Oscar René Valdés Casillas por confiar en mí y brindarme todo su apoyo, sin usted no hubiera podido titularme pues fue ese barco que nos llevó a tierra firme cuando quedamos por un momento perdidos.

A mis Amigos

Paty Cansino, Vonny, Mushu, Liz, Rosita, Víctor, Daniel, Paz, Jahir, Cesar, Alexis, Sol, Shirly, Betiko, Cris, Fer y Pao, Aún faltan más pero no acabaría de mencionar a todos, pero gracias por permitirme ser parte de su historia, por las aventuras y alegrías vividas, por brindándome su confianza y apoyo, a mi compañero de tesis David, vino a ser parte esencial en mi vida al cual se le estima mucho.

Por ello hoy quiero agradecerles de todo corazón el haberme dado esta oportunidad de superarme, andar paso a paso con cada uno de ustedes en esta carrera llamada ingeniería, por alentarme a seguir adelante y así, llegar a la meta que hoy culmina.

Con todo mi cariño, respeto y admiración.

Y Aquí no termina todo, pues vamos por más, así que con la ayuda de Dios, esta historia continuará…

David Emmanuel Contreras Barragán.

Esta tesis es trabajo de años, no tendría palabras para describir el agradecer a toda la gente que pasó a lo largo de esta carrera, trataré de ser lo más breve posible y tratar que no se me pase nadie, quiero empezar por aquellos profesores que han marcado mi vida que poco o mucho me han enseñado algo nuevo y aprendí de ellos, entraría primero la academia de telecomunicaciones, tuve momentos buenos, momentos malos. Oscar Valdez, Paty Hong, Song, Ivan, Ocampo, Manuel y Aranda me enseñaron que aunque tienes un hijo bueno o malo pues lo vas apoyar quieras o no porque al final es tu deber, quiero hacer otra mención al profesor Daniel aprendí que aunque no estés del todo bien, siempre habrá quien te eche la mano y te haga ver los errores, al profesor Estrada que él me dio la bienvenida a esta casa de estudios llamada UACM y que con un pequeño reconocimiento, cómo regalarte un libro puede incentivarte a seguir aprendiendo y conocer más, por ultimo está Velásquez y Tejinder, agradecerles por ser parte del jurado y acompañarme a cerrar este ciclo.

Después están mis maestros de vida José Luis, Israel y Zopi cada uno de ellos estuvieron en diferentes etapas de mi vida José Luis me enseño lo duro que es la vida, Zopi la paciencia y serenidad ante la vida y por último Israel el equilibrio en la vida, estos tres maestros marcaron un antes y un después antes de conocerlos, después el equipo de Fox Fitness donde me enseñaron el gusto por la vida y el por qué vale la pena vivirla, y por último quiero hacer una mención al profesor Macías, el me enseñó que ser el apestado no es tan malo y que la vida puede dar muchas vueltas de un día para otro.

Luego viene las menciones especiales gracias amigos que conocí en la UACM Fila, Pedro, Magda, Cristian, Juan, Valeria, Diana, Apa y otras personas más que conocí que si escribo todos los nombres no terminaría cada uno tiene su historia son especiales para mí.

Después esta mi familia que ellos vivieron todo mi proceso, tíos, tías, primos, primas familia Barragán y parte de la Contreras ya que no todos estuvieron en el proceso, mi carnalazo Luis Edgar que ha estado ahí echando relajo y también alentándome a lo que hago, a mis hermanos Mitzi y Raziel que ellos son un gran apoyo en mi vida y estuvieron en todo momento tanto buenos como malos, a mi compañera Miriam ella vino a cambiar parte de mi mentalidad e igual a creer en mi cuando yo ni siquiera creía en mí, se le estima mucho y por último a mis padres David y Lourdes ya que sin ellos yo no sería nada de lo que soy ahora, creo que ellos siempre han querido lo mejor para mí, no son los padres perfectos ni los mejores pero ellos sí sé que aman plenamente y bueno no quiero dejar fuera a Julián este niño me enseñó a querer más y tener paciencia.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1

ANTECEDENTES

1.1 Redes de computadoras

Una red de computadoras se define como un conjunto de computadoras autónomas conectadas entre sí mediante un medio de transmisión, con el objetivo de que el uso de las redes de computadoras en las empresas, las redes domésticas y los usuarios móviles, tengan la facilidad de compartir recursos e información y que estos estén disponibles cuando se requieran. Los recursos pueden incluir archivos, aplicaciones, impresoras y servicios, mientras que la información puede consistir en mensajes, correos electrónicos y datos diversos. [1]

La historia de las redes de computadoras comienza a finales de los años 1950, durante la Guerra Fría. En 1957, tras el lanzamiento del satélite Sputnik por la Unión Soviética, Estados Unidos se vio superado en la carrera espacial. El presidente Eisenhower, preocupado por la descoordinación entre las ramas militares en la investigación, creó la Agencia de Proyectos de Investigación Avanzados (ARPA, por sus siglas en inglés: the Advanced Research Projects Agency) para centralizar los esfuerzos de defensa. ARPA no tenía laboratorios propios, sino que financiaba investigaciones en universidades y empresas. A medida que la Guerra Fría avanzaba, el Departamento de Defensa de Estados Unidos (DoD, por sus siglas en inglés: United States Department of Defense) buscaba una red de comando y control que pudiera sobrevivir a un ataque nuclear, ya que las comunicaciones militares dependían de la vulnerable red telefónica pública. [1]

En 1967, Larry Roberts, director de ARPA, comenzó a explorar la creación de redes de computadoras para permitir acceso remoto. Tras consultar a expertos, decidió seguir la idea de Wesley Clark de construir una subred de conmutación de paquetes, donde cada computadora

estaría conectada a su propio enrutador. A pesar de las dudas iniciales, Roberts presentó la idea en una conferencia en 1967. Allí descubrió que, en Inglaterra, Donald Davies y su equipo en el Laboratorio Nacional de Física (NPL, por sus siglas en inglés: National Physics Laboratory) ya habían realizado un sistema de conmutación de paquetes a pequeña escala, demostrando que la tecnología funcionaba. Esto convenció a Roberts de avanzar con el proyecto que eventualmente se convertiría en la Red de Agencias de Proyectos de Investigación Avanzada (ARPANET, por sus siglas en inglés: The Network of advanced Research Project Agencies) la primera red de computadoras y precursora del Internet moderno. [1], [2]

1.2 Modelo de Referencia OSI (MR-OSI)

"Este modelo conocido por Interconexión de Sistemas Abiertos (OSI, por sus siglas en inglés: Open Systems Interconnection) cuya actividad vio la luz a principios de 1977 y obtuvo el grado definitivo de estándar internacional en 1983, trata de establecer las bases para la definición de protocolos de comunicación entre sistemas informáticos. Su principal objetivo es la interconexión de sistemas de diferentes fabricantes es decir de sistemas abiertos. Por ello OSI constituye un marco coordinación de las actividades de normalización en los sistemas de telecomunicaciones e información" [2]

El principio del MR-OSI se realizó a mediado de los años setenta cuando debido al interés sobre el diseño de bases de datos distribuidas, surgió la necesidad de una arquitectura de comunicaciones estructurada y distribuida. El grupo Canepa, recibe el nombre por Hubert Canepa, ingeniero y ejecutivo activo, junto con su grupo analizó alguna de las soluciones existentes en aquel momento, entre ellas la Arquitectura de Red de Sistemas (SNA, por sus siglas en inglés: Systems Network Architecture) de Máquinas de Negocios Internacionales (IBM, por sus siglas en inglés: International Business Machines) y los trabajos sobre protocolos de ARPANET. El resultado de este trabajo dio lugar en 1977 a una arquitectura de siete niveles conocida internamente como Arquitectura de Sistemas Distribuidos (ASD, por sus siglas en inglés: Distributed System Architecture). Al mismo tiempo, el instituto británico de Estandarización propuso a ISO[1] la necesidad de un estándar de arquitectura que soportara la

[1]Organización Internacional de Normalización (ISO, por sus siglas en inglés: International Organization for Standardization) es un órgano cuya función es la de crear normas de carácter internacional. [2]

definición de la infraestructura de los procesos distribuidos. ISO constituyo un comité sobre interconexión de sistemas abiertos (Comité técnico 97, Subcomité 16), siendo ANSI[2] el encargado del desarrollo de las propuestas. [2]

"En 1978 se llegó a un consenso entre las propuestas de ANSI, en colaboración con el grupo Canepa, y las que presentó Honeywell. Resultado del acuerdo fue la elección de una arquitectura por niveles y la consideración de que esta satisfaría la mayoría de los requerimientos exigibles a la interconexión de sistemas abiertos, así como el reconocimiento de la capacidad de expansión del modelo para adecuarse a los requerimientos futuros. Ese mismo año se publicó la versión provisional del modelo. En junio de 1979 la siguiente versión, sin apenas cambios, paso a convertirse en estándar. El modelo OSI definitivo es básicamente el mismo modelo DSA desarrollado en 1977". [2]

a) Definiciones de capas de Modelo OSI

El modelo OSI, es un marco conceptual que estandariza las funciones de una red de telecomunicaciones o un sistema de computación en siete capas distintas. Fue desarrollado por la Organización Internacional de Normalización (ISO) en 1984 y sigue siendo una referencia en el campo de las redes y las comunicaciones. [2], [3]

La especificación ISO se basó en definir conjunto de capas y los servicios que cada una debía realizar. Debía implicar el suficiente número de capas para que su complejidad fuera pequeña y tampoco debía ser elevado el número de capas. El modelo de referencia resultante tiene siete capas y se explica la forma detallada en la figura 1.1. La principal motivación para el desarrollo de este modelo era proporcionar una normalización. Dentro de este modelo en cada capa se desarrolla uno o más protocolos, define una función de cada etapa y lo simplifica. [2], [3]

Se utiliza el principio de ocultación de la información. Las capas inferiores abordan ciertos detalles de tal manera que las superiores sean ajenas, cada capa tiene una tarea en específico. Forma más específica la normalización requerida de cada capa, tiene tres elementos: [2], [3]

[2] Instituto Nacional Estadounidense de Estándares (ANSI, por sus siglas en inglés: American Nacional Standards Institute): es una organización encargada de supervisar el desarrollo de normas para los servicios, productos, procesos y sistemas en los Estados Unidos. [3]

- **Especificación del protocolo:** El protocolo se debe especificar con precisión ya que están implicados dos sistemas abiertos diferentes, incluye el formato de unidad de datos del protocolo, la secuencia permitida de la Unidad de Datos de Protocolo (PDU, por sus siglas en inglés: Protocol Data Unit), la semántica de todos los datos.

- **Definición del servicio:** Además de los protocolos o protocolo que se opera en una capa, se necesitan normalizaciones para los servicios de cada capa, La definición de los servicios es equivalente a una descripción funcional, define qué servicios se están proporcionando, pero no cómo se están proporcionando.

- **Direccionamiento:** Cada capa suministra servicios a las entidades en la capa superior. Las entidades se identifican mediante un Punto de Acceso al Servicio (SAP, por sus siglas en inglés: Service Access Point) y un Punto de Acceso al servicio de red (NSAP, por sus siglas en inglés: Network Service Access Point).

Capa 1: Capa Física

- **Función**: Esta capa se encarga de la transmisión de bits puros a través de un medio físico. Define las características físicas de los dispositivos y medios de transmisión, como voltajes, niveles de señal, velocidad de bits, y la modulación.
- **Componentes y Ejemplos**:

  Medios de transmisión: Cables (cobre, fibra óptica), ondas de radio.

  Dispositivos: Repetidores, módems.

Capa 2: Capa de Enlace de Datos

- **Función**: Proporciona un enlace de datos confiable entre dos nodos directamente conectados, detectando y corrigiendo errores que pueden ocurrir en la capa física. Divide los datos en tramas y maneja las direcciones físicas, el Control de Acceso a Medios (MAC, por sus siglas en inglés: Media Access Control).

- **Protocolos**: Protocolo de Resolución de Direcciones (ARP, por sus siglas en inglés: Address Resolution Protocol)

Capa 3: Capa de Red

- **Función**: Determina la mejor ruta para los datos a través de una red y maneja las direcciones lógicas (IP addresses). Esta capa se encarga de la entrega de paquetes desde el origen hasta el destino, incluso si los dos no están en la misma red.
- **Protocolos**: Protocolo de Internet (IP, por sus siglas en inglés: Internet Protocol), Protocolo de Mensajes de Control de Internet (ICMP, por sus siglas en inglés: Internet Control Message Protocol).

Capa 4: Capa de Transporte

- **Función**: Proporciona transferencia de datos confiable y no confiable entre dos hosts (un host es un dispositivo o sistema que está conectado a una red y que puede enviar, recibir o procesar información). Se encarga de la segmentación, reensamblaje de datos y control de flujo. Proporciona reconocimiento y retransmisión de datos en caso de error.
- **Protocolos**: Protocolo de Control de Transmisión (TCP, por sus siglas en inglés: Transmission Control Protocol para transferencia confiable, Protocolo de Datagramas de Usuario (UDP, User Datagram Protocol) para transferencia no confiable.

Capa 5: Capa de Sesión

- **Función**: Establece, gestiona y termina las sesiones entre aplicaciones. Maneja el control de diálogo, la sincronización y la recuperación de datos.
- **Protocolos e Interfaz de Programación de Aplicaciones (APIs, por sus siglas en inglés: Application Programming Interface):** Sistema Básico de Entrada/Salida de Red (NetBIOS, por siglas en inglés: Network Basic Input/Output System), Llamada a Procedimiento Remoto (RPC, por sus siglas en inglés: Remote Procedure Call), Protocolo

de Autenticación de Contraseña (PAP, por sus siglas en inglés: Password Authentication Protocol).

Capa 6: Capa de Presentación

- **Función**: Traduce los datos entre el formato de la red y el formato que la aplicación puede procesar. Esta capa maneja la encriptación, la compresión de datos y la conversión de formatos.
- **Formatos de Datos**: JPEG[3], GIF[4], TIFF[5] (imágenes), ASCII[6], EBCDIC[7] (texto).
- **Protocolos**: SSL/TLS[8] (seguridad de capa de sockets seguros y capa de transporte), códecs[9] de audio y video.

Capa 7: Capa de Aplicación

- **Función**: Proporciona servicios de red directamente a las aplicaciones del usuario final. Es la capa más cercana al usuario y maneja las interacciones con el software de la aplicación.

---

[3] Grupo de Expertos en Fotografía (JPEG, por sus siglas en inglés: Joint photographic Experts Group): Es un formato de comprensión de imágenes desarrollado por este grupo. [2]

[4] Formato de Intercambio de Gráficos (GIF, por sus siglas: Graphics Exchange Format): Formato de imagen desarrollado por CompuServe que admite gráficos de baja resolución y permite animaciones. [2]

[5] Formato de Archivo de Imagen Etiquetado (TIFF, por sus siglas en inglés: Tagged Image File Format): Formato de Imagen de Alta Calidad, utilizado en la industria gráfica y editorial. [2]

[6] Código Estándar Americano para el intercambio de información (ASCCI, por sus siglas en inglés: American Standard Code for Information Interchange): Es un código de caracteres utilizado en sistemas informáticos para representar letras, números y otros símbolos. [2]

[7] Código de Intercambio Decimal de Código Binario (EBCDIC, por sus siglas en inglés: Extended Binary Code Decimal Exrchange Code): Es un sistema de codificación de caracteres desarrollado por IBM. [3]

[8] Capa de sockets seguros (SSL, por sus siglas en inglés: Secure Sockets Layer) y Capa de transporte seguro (TLS, por sus siglas en inglés: Transport Layer Security): Ambos son protocolos de seguridad diseñados para proteger la comunicación en internet al cifrar los datos que se envían entre un cliente y un servidor web. [3]

[9] Códecs es una abreviatura de las palabras en inglés "coder-decoder". Se refiere a programas que codifican (comprimen) y decodifican (descomprimen) datos digitales, en el contexto de audio y video. Los códecs son esenciales para la transmisión, almacenamiento y reproducción eficiente de contenido multimedia. [3]

- **Protocolos**: Protocolo de Transferencia de Hipertexto (HTTP por sus siglas en inglés: Hypertext Transfer Protocol) para la web, Protocolo de Transferencia de Ficheros (FTP, por sus siglas en inglés: File Transfer Protocol) para transferencia de archivos, Protocolo Simple de Transferencia de Correo (SMTP, por sus siglas en inglés: Simple Mail Transfer Protocol) para correo electrónico, Sistema de Nombres de Dominio (DNS, por sus siglas en inglés: Domain Name System) para resolución de nombres.

Cada una de estas capas interactúa con la capa inmediatamente superior e inferior para proporcionar una comunicación coherente y eficiente a través de la red. En la figura 1.1 se muestra una imagen detallada de este modelo. [2], [3]

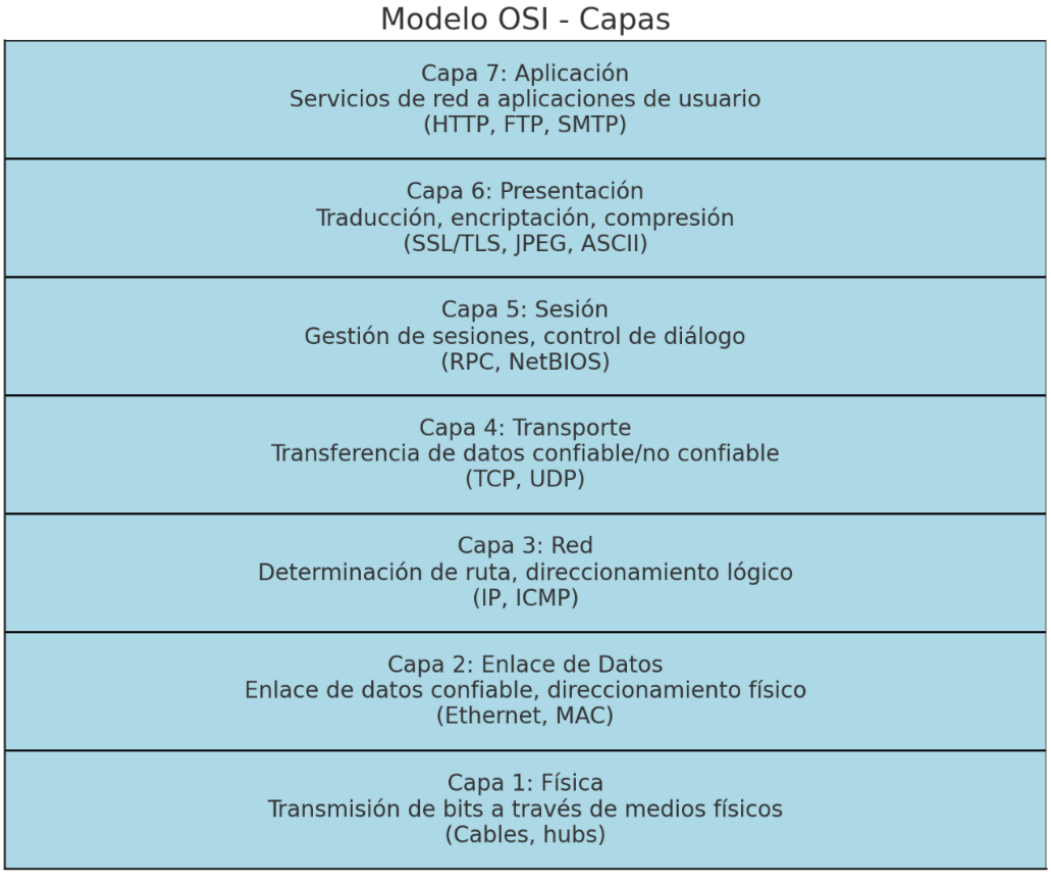

Figura 1.1 Capas del Modelo OSI. [3]

1.3 Dispositivos de red

"Un dispositivo de red es un componente de hardware el cual se utiliza para conectar computadoras, impresoras, teléfonos, y otros dispositivos electrónicos dentro de una red de área local (LAN, por sus siglas en inglés: Local Area Networks), una red de área amplia (WAN, por sus siglas en inglés: Wide Area Networsk) o en redes más grandes, como Internet. Estos dispositivos permiten la transmisión de datos entre diferentes equipos y facilitan la comunicación y el intercambio de información" [1], [3]

A continuación, se enlistan algunos dispositivos de red.

1. **Router (Dispositivo de encaminamiento)**: Es un dispositivo de red que conecta dos redes de computadoras usando una arquitectura y un protocolo de interconexión común, operando en la capa 3 del modelo OSI.
2. **Switch (Conmutador)**: El switch es un dispositivo de red que por lo general contiene de 4 a 48 puertos, enviando tramas solo al puerto de destino correspondiente, utilizando un plano posterior de alta velocidad para reenviar las tramas y deduce las direcciones de los puertos a los que están destinadas. El switch opera en la capa 2 del modelo OSI.
3. **Hub (Concentrador)**: Un dispositivo que conecta múltiples computadoras en una red. A diferencia de un switch, un hub envía datos a todos los dispositivos conectados, lo que puede generar tráfico innecesario. Los cuales se clasifican como elementos de la capa 1 del modelo OSI operando en la capa física.
4. **Access Point (Punto de Acceso)**: Un dispositivo que permite a dispositivos inalámbricos conectarse a una red cableada. Los puntos de acceso en redes inalámbricas. Su principal función y gestión están centradas en la capa 2 del modelo OSI.
5. **Firewall (Servidores de seguridad)**: Actúa como un filtro de paquetes, inspeccionando a los paquetes entrantes y salientes, controlando el tráfico de red para accesos no autorizados o ciberataques. Opera en la capa 3del modelo OSI
6. **Bridge (Puente)**: Un puente es un sistema de interconexión entre dos redes LAN que usan el mismo protocolo, filtrando y reenviando paquetes entre ellas sin modificar su contenido. Es la base del switch y opera en la capa 2 del modelo OSI.

7. **Gateway (Puerta de enlace):** Es una puerta de enlace que puede operar en diversas capas del modelo OSI, desde la capa 3 hasta la capa 7. Dependiendo de su función y propósito, el Gateway para interconectar redes o aplicaciones con arquitecturas y protocolos diferentes.
8. **Tarjeta de Interfaz de Red (NIC, por sus siglas en inglés: Network Interface Card)**: Es un hardware dedicado que ejecuta parte de los procesos de la capa física y de la capa de enlace de datos en una red. El resto de estos procesos, junto con los de la capa de red, se ejecutan en la Unidad Central de Procesamiento (CPU) principal como parte del sistema operativo, a menudo en forma de un controlador de dispositivo. La NIC facilita la comunicación entre estas capas y destaca la independencia entre ellas.

Estos dispositivos de red son necesarios para establecer, mantener y proteger la comunicación entre los dispositivos en una red, garantizando una transferencia de datos eficiente y segura. [1], [3], [4]

1.4 Protocolo

Los protocolos son formatos de comunicación normalizados para lenguaje común entre sistemas y fabricantes que rigen el funcionamiento de unidades funcionales para realizar la comunicación, Estos protocolos definen cómo los datos deben ser transmitidos, dirigidos y recibidos en una red, asegurando que los diferentes dispositivos puedan comunicarse de manera efectiva y sin errores. Cada capa del modelo OSI tiene sus protocolos específicos que manejan aspectos particulares de la comunicación de red. [5]

Aspectos que un protocolo puede definir:

- **Formato de los mensajes**: Cómo se estructuran los paquetes de datos.
- **Secuencia de mensajes**: El orden en que deben enviarse y recibirse los datos.
- **Control de errores**: Cómo manejar los errores que puedan surgir durante la transmisión.
- **Control de flujo**: Cómo gestionar la velocidad de envío de datos para evitar saturación.

Ejemplos de protocolos:

- **HTTP**: Protocolo utilizado para la transferencia de datos en la web.

- **TCP/IP**: Protocolo utilizado para la transmisión de datos en Internet.
- **FTP**: Protocolo para la transferencia de archivos.

Un protocolo es esencialmente un acuerdo entre las entidades que se comunican en una red, que especifica cómo deben intercambiar datos para que la comunicación sea exitosa y comprensible por ambas partes. [5]

1.5 Tipo de Redes

Redes de Área Local (LAN)

- **Descripción**: Esta red puede incluir a dos computadoras en una vivienda privada o a varios miles de dispositivos en una empresa, instituciones públicas como administraciones, colegios o universidades. La transmisión de datos se realiza de manera electrónica a través de cables de cobre o mediante fibra óptica. [6], [7]
- **Alcance**: Limitadas a un área geográfica pequeña, como un solo edificio o un campus universitario. En la figura 1.2 se muestra la distancia del área de la red.
- **Características**: Velocidad de transmisión alta (generalmente entre 100 Mbps y 1 Gbps), baja latencia, y típicamente propiedad de la misma organización.
- **Tecnologías Comunes**: Ethernet, Wi-Fi.
- **Estándar para la red LAN**: IEEE802.11 (Wi-Fi) y IEEE802.3 (Ethernet)

Redes de Área Amplia (WAN)

- **Descripción**: Es una red que conecta por ejemplo las oficinas que se encuentran en diferentes ciudades, donde las computadoras ejecutan aplicaciones. En una red WAN, la subred tiene dos componentes principales: las líneas de transmisión, que mueven bits entre máquinas, y los elementos de conmutación. Estas líneas pueden ser de cobre, fibra óptica o utilizar enlaces de radio. [6], [7]
- **Alcance**: Cubren áreas geográficas extensas, como un país o un continente. En la figura 1.2 se describe la distancia que puede abarcar la red.

- **Características**: Menor velocidad de transmisión comparada con LAN y MAN, mayores latencias debido a la distancia.
- **Ejemplos**: Internet, redes corporativas globales.

En la figura 1.2 se muestra la clasificación de las diversas distancias de áreas de red entre procesadores y ubicación. [1]

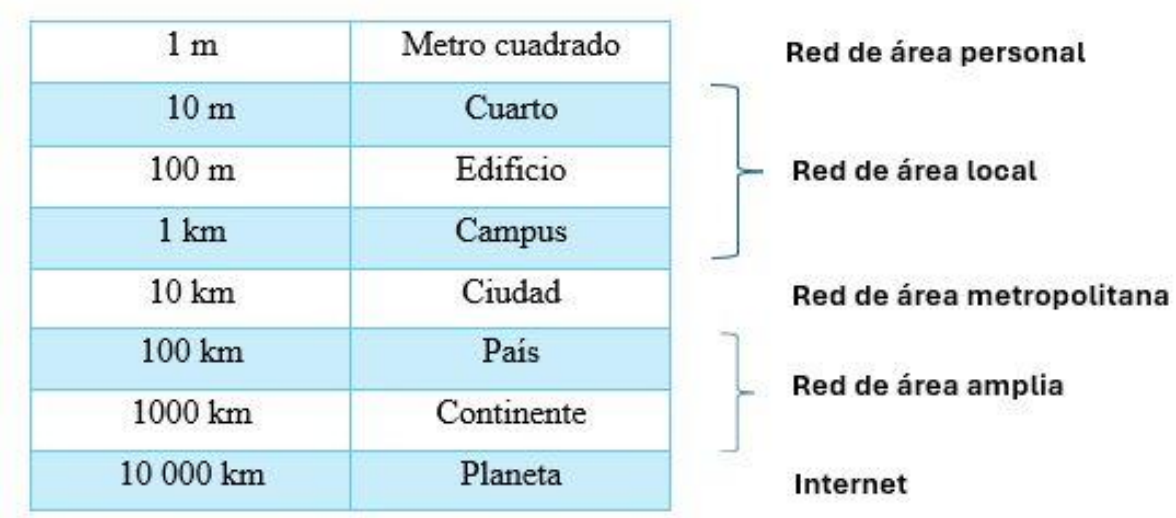

Figura 1.2 Clasificación de los procesadores interconectados con base en la escala. [1]

1.6 Internet

"Internet se describe como una red de diversas redes locales y globales, incluyendo redes públicas, privadas, académicas, comerciales o gubernamentales. Estas redes se conectan y permiten la conexión entre dispositivos a través de protocolos estándar como TCP/IP, con el IP como eje central. Esto permite el intercambio de información a escala mundial. Su notable evolución y adaptabilidad se deben a que no fue planificada ni controlada por una sola entidad, lo que ha sido importante para su crecimiento y relevancia mundial" [3], [7]

1.7 Topologías

En el contexto de una red de comunicaciones el término topología se refiere al modo de conectar entre sí los puntos finales o estaciones de una red de computadoras. Las topologías comunes en redes LAN son bus, árbol, anillo y estrella. [2], [7]

Topología de Bus

- **Descripción**: Esta topología se caracteriza por el uso de un medio multipunto, todas las estaciones se conectan directamente a un bus mediante tomas de conexión, como se muestra en la figura 1.3a, permitiendo la transmisión y recepción de datos en full-duplex. Las transmisiones se propagan en ambas direcciones y son recibidas por todas las estaciones.
- **Ventajas**: Simplicidad, bajo costo.
- **Desventajas**: Dificultad para aislar fallas, rendimiento disminuye con el aumento de dispositivos.

Topología de Árbol

- **Descripción**: El medio de transmisión es un cable ramificado sin ciclos cerrados que empieza en el punto conocido como raíz y cada uno de ellos puede presentar ramificaciones y cada rama puede disponer de ramas adicionales. Una combinación de topologías de estrella y bus, con una estructura jerárquica.
- **Ventajas**: Facilidad para expandir y segmentar la red.
- **Desventajas**: Complejidad de configuración, si dos estaciones intentan transmitir simultáneamente sus señales se superpondrán y serán erróneos.

Topología de Anillo

- **Descripción**: La red consta de un conjunto de repetidores unidos por enlaces punto a punto formando un ciclo cerrado, la figura 1.3c muestra que el enlace es unidireccional circulando alrededor del anillo en sentido de las agujas del reloj.

- **Ventajas**: Transmisión de datos sencilla y ordenada.
- **Desventajas**: Un fallo en una sola computadora puede afectar a toda la red.

Topología de Estrella

- **Descripción**: Cada estación está conectada a un nodo central o concentrador (Hub) común a través de dos enlaces punto a punto, uno para transmisión y otro para recepción.
- **Ventajas**: Facilidad para agregar nuevos dispositivos, el fallo de una computadora no afecta a la red.
- **Desventajas**: El fallo del dispositivo central puede desactivar toda la red.

En la figura 1.3 se muestra el ejemplo de cada una de las topologías LAN. [2], [7]

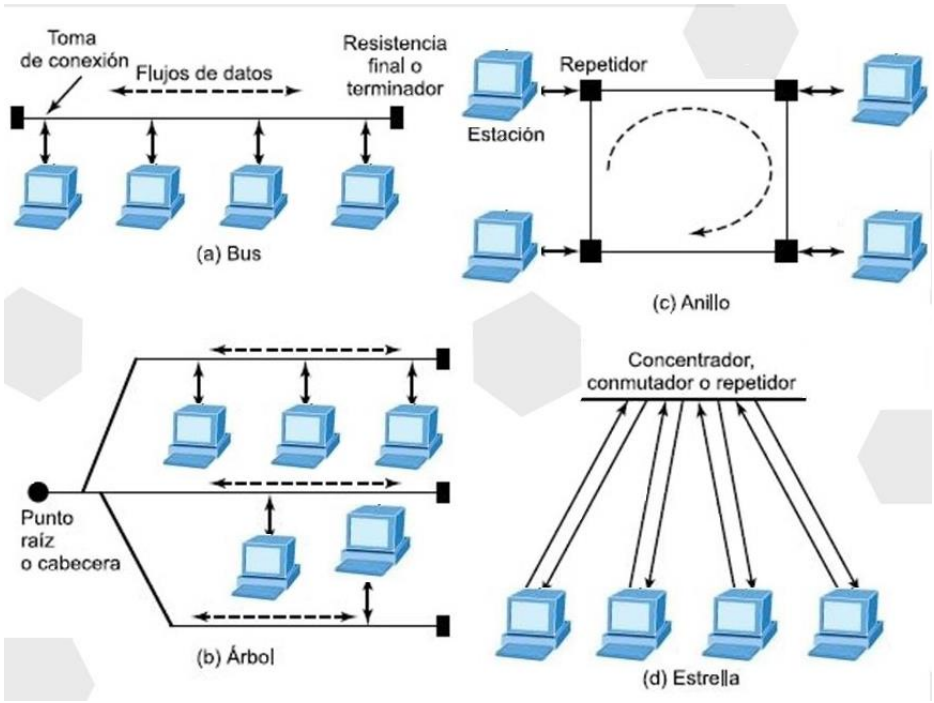

Figura 1.3 Topologías LAN. [3]

1.8 Suite de Protocolos: Protocolo de Control de Transmisión/Protocolo de Internet (TCP/IP, Transmission Control Protocol/Internet Protocol)

En 1982 se especificó un nuevo conjunto de protocolos para ARPANET que fue referenciada por TCP/IP. Se suministraron implementaciones con las versiones 4.1 y 4.2 BSD[10] de UNIX, [11]lo cual facilito su expansión. [1], [5]

En 1983 TCP/IP se adoptó como estándar en ARPANET. A su vez, una segunda red llamada, Red Militar (MILNET, por sus siglas en inglés: Military Network) dedicada exclusivamente a aspecto militares, se separó de ARPANET, y así comenzó a formarse la red INTERNET, actualmente TCP/IP se considera como el conjunto de protocolos abiertos, con lo que se ha convertido en un estándar facto, el cual es una práctica o tecnología que se convierte en la norma por su uso generalizado, sin ser aprobada oficialmente por una organización y es aceptado por mucha gente o empresas. [1], [5]

a) Funcionamiento de capas del protocolo TCP/IP

TCP/IP es un modelo de comunicación en capas que facilita la transmisión de datos entre dispositivos en redes, proporcionando un conjunto de reglas estándar para que diferentes sistemas puedan intercambiar información. Está compuesto por dos protocolos principales: [1], [5], [7]

1. **TCP (Transmission Control Protocol)**: Responsable de garantizar la confiabilidad en la transmisión de datos. Divide los datos en segmentos, los envía, controla su correcta llegada y los reordena en el destino si es necesario.
2. **IP (Internet Protocol)**: Protocolo encargado de enrutar los paquetes de datos a través de las redes hacia su destino final.

[10] Distribución de Software de Berkeley (BSD, por sus siglas en inglés: Berkeley Software Distribution): Es el nombre de las distribuciones de código fuente de la universidad de California, Berkeley, de código abierto. BSD se encarga de la programación de procesos, gestión de memoria, multiprocesos simétricos, controladores de dispositivos, etc. [9]

[11] Sistema de computación e información Uniplexed (UNIX, por sus siglas en inglés: Uniplexed Information and Computing System): Se encuentra registrado como marca y protege por derechos de autor a todos los sistemas Unix, siendo un sistema operativo de los años 70 de código abierto, sus características principales son: portabilidad, multiusuario y multitarea, eficiencia, alta seguridad y buen desempeño en tareas de red [10]

El modelo TCP/IP sigue un enfoque en capas, donde cada capa tiene funciones específicas:

1. Capa Física

- **Función**: Se encarga de la transmisión de bits sin procesar a través del medio físico de comunicación, ya sea por cables, fibra óptica o señales inalámbricas. Define las características eléctricas y físicas del medio de transmisión, como el voltaje o las frecuencias de las señales.
- **Propósito:** Asegurar que los datos (en forma de bits) se transmitan correctamente entre dispositivos conectados físicamente en una red.

2. Capa de Enlace de Datos

- **Función:** Gestiona la transmisión de tramas de datos entre dos nodos directamente conectados en la misma red física (como dentro de una LAN). Se encarga de la detección y corrección de errores a nivel de tramas y de asegurar que los datos se transmitan de manera confiable en una red local.
- **Protocolos comunes:** Ethernet, Wi-Fi, PPP (Protocolo Punto a Punto).
- **Propósito:** Proporcionar un canal libre de errores entre dos dispositivos directamente conectados.

3. Capa de Red

- **Función:** Es el responsable del direccionamiento, se ocupa del enrutamiento de los paquetes de datos desde el dispositivo emisor hasta el destino, incluso si están en redes diferentes. En esta capa, se usa el protocolo IP (Internet Protocol), que permite que los datos viajen a través de múltiples redes interconectadas.
- **Protocolo principal:** IP (Internet Protocol).
- **Propósito:** Asegurar que los datos lleguen a su destino final, gestionando las direcciones IP y el enrutamiento entre redes.

4. Capa de Transporte

- **Función:** Proporciona una transmisión confiable o rápida de datos entre dos sistemas, según el protocolo utilizado. Se encarga de la segmentación de los datos y de asegurar que estos lleguen completos y en el orden correcto. También gestiona el control de flujo y la recuperación de errores a nivel de segmento.
- **Protocolos principales:**
  - **TCP (Transmission Control Protocol):** Ofrece una transmisión confiable de datos.
  - **UDP (User Datagram Protocol):** Protocolo más rápido, pero no garantiza la confiabilidad ni el orden.
- **Propósito:** Garantizar la entrega confiable (TCP) o rápida (UDP) de los datos entre el emisor y el receptor.

5. Capa de Aplicación:

- **Función:** Proporciona los servicios que utilizan las aplicaciones de usuario para comunicarse con la red. Esta capa contiene protocolos que permiten diferentes servicios, como el acceso web, envío de correos electrónicos, o transferencias de archivos.
- **Protocolos comunes:** HTTP (para navegación web), FTP (para transferencia de archivos), SMTP (para correos electrónicos), DNS (para resolución de nombres).
- **Propósito:** Facilitar la interacción de las aplicaciones del usuario con la red a través de protocolos específicos para cada tarea.

El modelo TCP/IP es un enfoque modular y robusto para la comunicación en red, donde cada capa tiene responsabilidades específicas y trabaja en conjunto para asegurar que los datos viajen desde una aplicación en un dispositivo a una aplicación en otro, de manera confiable y eficiente. Esta estructura ha permitido la expansión y el éxito de Internet, proporcionando un marco flexible que puede adaptarse a nuevas tecnologías y aplicaciones. [1], [5], [7]

1.9 El Protocolo de Transferencia de Hipertexto (HTTP, Hypertext Transfer Protocol)

HTTP fue desarrollado alrededor de 1989 por Tim Berners-Lee, de origen inglés, y Robert Cailliau, de Bélgica, mientras trabajaban en la Organización Europea para la Investigación Nuclear (CERN, por sus siglas en inglés: European Organization for Nuclear Research) en Ginebra, Suiza. Fue lanzada en 1992 como parte de la creación de la World Wide Web (WWW). Berners-Lee ideó HTTP para permitir la transferencia de hipertextos, facilitando la comunicación entre servidores y clientes web. El protocolo fue diseñado para ser sencillo y eficiente, soportando la conexión de dispositivos en una red mediante el uso de métodos como GET y POST para intercambiar datos. HTTP se convirtió en un estándar para la navegación web. [2], [10]

Es la base para la comunicación entre un cliente (generalmente un navegador web) y un servidor web. HTTP se basa en el modelo cliente-servidor, donde un cliente envía solicitudes al servidor para solicitar recursos web y el servidor responde a esas solicitudes proporcionando los recursos solicitados, como páginas web, imágenes, archivos de estilo o scripts, etc., o en dado caso, informando errores. [2], [10]

a) Solicitud De Comentarios (RFC, Request for Comments) referente al HTTP

"RFC, es un documento publicado por el Grupo de trabajo de ingeniería de Internet (IETF, por sus siglas en inglés: Internet Engineering Task Force) para la revisión, desarrollo y estandarización de los protocolos de Internet, como HTTP y TCP/IP. Cada RFC es una propuesta o estándar revisado por la comunidad técnica. HTTP/1.1 está definido en el RFC 2616, publicado en 1999, y describe detalles técnicos del protocolo, incluyendo métodos como GET[12] y POST[13]. En 2015, se introdujo HTTP/2, RFC 7540 mejorando la eficiencia y reduciendo la latencia al optimizar la transmisión de datos y permitir múltiples intercambios simultáneos en una misma conexión. Comparte los mismos números de puerto predeterminados: 80 para

[12] GET (Obtener): Este método se utiliza para solicitar datos desde el servidor sin modificar nada. Por ejemplo, en una página web, el navegador envía una solicitud GET para obtener el contenido de la página web. [12]

[13] POST (Enviar): Este método se usa para enviar datos al servidor con el fin de procesarlos. Por ejemplo, un formulario de registro, los datos enviados pueden modificar o actualizar la información en el servidor. [12]

URI[14]o URL[15] "http" y 443 para las URL o URI "https". Como resultado, las implementaciones procesan solicitudes de URI o URL de recursos de destino como "http://example.org/foo" o "https://example.com/bar". La última actualización se hizo el 21 de enero de 2020". [10]

b) Características del protocolo HTTP:

1. **Protocolo sin estado:** Lo que significa que cada solicitud del cliente es independiente de las solicitudes anteriores. El servidor no mantiene información sobre el estado de las conexiones, lo que simplifica la implementación y permite que las solicitudes sean procesadas de manera independiente. [1], [10]
2. **Métodos de solicitud:** Define varios métodos de solicitud, los más comunes son GET (Leer una página web), HEAD (Leer el encabezado de una página web), POST (Adjuntar a una página web), PUT (Almacenar una página web), DELETE (Eliminar la página web) y otros. Estos métodos indican la acción que se debe realizar en el recurso solicitado. [1], [10]
3. **URL (Localizador uniforme de recursos):** Las solicitudes incluyen una URL que identifica el recurso solicitado. [1], [10]
4. **Respuestas del servidor:** El servidor responde a las solicitudes con códigos de estado que indican si la solicitud se procesó correctamente o si hubo algún problema. Algunos códigos de estado comunes incluyen 200 OK (solicitud exitosa), 404 Not Found (recurso no encontrado) y 500 Internal Server Error (error interno del servidor). [1], [10]
5. **Protocolo basado en texto**: Utiliza un formato de mensaje basado en texto legible por humanos para las solicitudes y respuestas. Esto facilita la depuración y la comprensión, pero también puede ser menos eficiente que otros protocolos binarios en términos de uso de ancho de banda. [1], [10]

[14] Identificador Uniforme de Recursos (URI, por sus siglas en inglés: Uniform Resource Identifier). Se refiere a una cadena de caracteres que identifica de manera única un recurso en internet, incluye localizadores como nombres. [13]

[15] Localizador Uniforme de Recursos (URL, por sus siglas en inglés; Uniform Resource Locator): Proporciona una ubicación exacta donde se puede acceder a un recurso en la web, incluye el protocolo HTTP o HTTPS, el dominio, la ruta y parámetros. [12]

1.10 Lenguaje de Marcas de Hipertexto (HTML, Hyper Text Markup Language)

HTML fue creado en 1991 por Tim Berners de la Organización Europea para la Investigación Nuclear conocida como el CERN, como un lenguaje sencillo para compartir documentos en la Web. Se trata de un estándar para la construcción de páginas web y permite definir la estructura y el significado de los elementos en un documento web. Se compone de etiquetas que envuelven el contenido y describen su función o apariencia, facilita la navegación entre documentos y recursos interconectados en Internet. En 1994, el Consorcio de la Red Mundial (W3C, por sus siglas en inglés: World Wide Web Consortium) inició la estandarización, lo que impulsó un crecimiento masivo en la cantidad de documentos web, garantizando su interpretación uniforme en cualquier plataforma, promoviendo la accesibilidad global. [1], [3], [10]

Algunas características incluyen:

1. **Estructura:** Define la estructura básica de una página web, que consta de elementos como encabezados, párrafos, listas, enlaces, imágenes y más.
2. **Hipervínculos:** Un hipervínculo es un elemento (texto, imagen, icono) en una página que enlaza a otra página o recurso en línea, facilitando la navegación entre diferentes contenidos.
3. **Multimedia:** Admite la inclusión de contenido multimedia, como imágenes, audio y vídeo, en una página web.
4. **Formularios:** Facilita la creación de formularios interactivos que los usuarios pueden completar y enviar.
5. **Estructura Semántica:** Con la introducción de HTML5, se enfatiza la importancia de proporcionar una estructura semántica, lo que significa que las etiquetas deben reflejar el significado y la función del contenido.

Algunos ejemplos de etiquetas utilizadas de manera semántica incluyen:

1. <header>: Utilizado para el encabezado de una página o sección.
2. <nav>: Para la navegación principal de un sitio web.
3. <main>: Defina el contenido principal de una página.

4. <article>: Empleado para contenido independiente y autónomo, como una publicación de blog.
5. <section>: Utilizado para agrupar contenido relacionado.
6. <aside>: Para contenido relacionado, pero no esencial.
7. <footer>: Para el pie de página de una página o sección.

Compatibilidad: Es un estándar ampliamente compatible que funciona en la mayoría de los navegadores web, lo que garantiza que las páginas web sean accesibles para una audiencia amplia. HTML se combina con otros lenguajes y tecnologías web, como CSS (Cascading Style Sheets) para el diseño y la presentación visual, y JavaScript para la interacción y la dinámica en la web. Juntos, estos elementos forman la base de la creación de sitios web y aplicaciones web interactivas. [1], [3], [10]

a) Páginas Web Estáticas

La base de la web radica en la transferencia de páginas desde el servidor al cliente. En su forma más básica, las páginas web son estáticas, lo que significa que son archivos almacenados en un servidor que se muestran de la misma manera cada vez que un cliente las accede. Sin embargo, estáticas no implica que sean inactivas; una página estática puede incluir elementos como videos. HTML es el lenguaje principal de la web, utilizado comúnmente para crear tanto páginas estáticas como dinámicas. [1], [10]

b) Páginas Web Dinámicas

Las páginas dinámicas son importantes porque ofrecen una experiencia interactiva y personalizada que no es posible con páginas estáticas. A diferencia de las páginas estáticas, que muestran siempre el mismo contenido. Las páginas web dinámicas permiten a los usuarios interactuar en tiempo real con aplicaciones web directamente desde el navegador para acceder a datos actualizados, como la disponibilidad de libros en una biblioteca o el estado de las acciones en la Bolsa. Los usuarios se benefician al no necesitar instalar programas adicionales,

ya que pueden acceder a sus datos desde cualquier computadora. Esta funcionalidad es posible gracias a programas que se ejecutan en el servidor, en el navegador, o en ambos, ofreciendo una experiencia interactiva y accesible en cualquier momento. [1], [10]

c) Web 2.0

La Web 2.0 marca una evolución respecto a la Web 1.0, Internet en su versión inicial era estático, con páginas en HTML que ofrecían información, pero sin interacción del usuario. La Web 2.0, Introduce una segunda generación de la Web que enfatiza la interacción, colaboración y participación del usuario. Permite a los usuarios generar y compartir contenido, transformando Internet de un banco de datos a un espacio social interactivo. [13]

El impacto en la interacción social, la Web 2.0 ha permitido el surgimiento de plataformas sociales como Wikipedia, YouTube y Facebook, que facilitan la interacción y el intercambio de contenido entre usuarios. Además, las empresas han comenzado a aprovechar estas herramientas para mejorar sus servicios. [13]

Los beneficios para el Marketing, la mayoría de las empresas usan herramientas de la Web 2.0, como blogs, wikis, y webconference, para adaptarse a la evolución del mercado. Así como las estrategias innovadoras que demuestran cómo las empresas utilizan la Web 2.0 para involucrar a los usuarios en la mejora de productos y servicios, aplicando estrategias de marketing basadas en la interacción directa con el cliente. [13]

La Web 2.0 ha transformado Internet en una plataforma dinámica de interacción y colaboración, beneficiando tanto a usuarios como a empresas al permitir una participación más activa y estrategias de marketing más efectivas. [13]

d) JavaScript y XML[16] Asincrónicos (AJAX, Asynchronous JavaScript and XML)

AJAX, es un conjunto de tecnologías que permite crear aplicaciones web altamente interactivas, similares a las aplicaciones de escritorio. Combina HTML y Hojas de Estilo en Cascada (CSS, por sus siglas en inglés: Cascading Style Sheets) para presentar contenido, el Modelo de Objetos de Documento (DOM) para modificar partes de una página sin recargarla, y XML para intercambiar datos estructurados con el servidor. La importancia de AJAX es su capacidad para realizar solicitudes asíncronas, lo que significa que la página no se bloquea mientras espera una respuesta del servidor, manteniendo la interfaz del usuario receptiva. JavaScript es el lenguaje que une todos estos componentes, permitiendo que las páginas web se actualicen dinámicamente en función de las interacciones del usuario sin necesidad de recargar la página completa. AJAX optimiza la experiencia del usuario al permitir que la información se actualice y se intercambie en segundo plano, sin interrumpir la interacción con la página web. Esto es especialmente útil en aplicaciones web como Google Maps o Gmail, donde la actualización constante de datos y la fluidez en la navegación son cruciales para la experiencia del usuario. [1]

1.11 Servidores WEB

Un servidor web es un software que opera en un sistema físico o virtual y tiene como función entregar contenido web a los usuarios a través de Internet. Cuando un usuario accede a una página web desde su navegador (como Chrome, Firefox o Safari), el proceso comienza con el navegador, analizando la URL, la cual interpreta la sección entre el protocolo "http://" y la siguiente barra diagonal como un nombre DNS o IP donde se encuentra el recurso que debe buscar. Una vez que obtiene la Ruta o ubicación exacta en el navegador, establece una conexión TCP con el puerto 80 de dicho servidor. [14]

Entonces, sigue un ciclo principal de pasos para procesar la solicitud del navegador:

1. **Acepta una conexión TCP** del cliente (el navegador).

[16] Lenguaje de Marcado Extensible (XML, por sus siglas en inglés: Extensible Markup Language): Es un formato basado en texto para representar la información de manera estructurada como: documentos, datos configuraciones, libros, facturas, etc. [2]

2. **Obtiene la ruta de la página**, que corresponde al nombre del archivo solicitado.
3. **Accede al archivo** en el disco.
4. **Transmite el contenido** del archivo al cliente.
5. **Libera la conexión TCP**.

Este ciclo permite que el navegador reciba el contenido solicitado, como un archivo HTML, una imagen o un script, y lo muestre en la pantalla del usuario.

Además de manejar solicitudes de páginas web simples o dinámicas, los servidores web son capaces de garantizar que la comunicación entre el servidor y el cliente sea segura, confiable y eficiente. La popularidad de diferentes servidores web varía según el contexto de uso, algunos son preferibles dependiendo del proyecto en el que se trabaje, por ejemplo, para manejar grandes volúmenes de datos, otros por compatibilidad con otras tecnologías o sistemas operativos, y otros por su facilidad de configuración y mantenimiento. Con la evolución de la web, se han adaptado para soportar nuevas tecnologías y paradigmas, como la visualización de contenido dinámico, las aplicaciones de tiempo real y la implementación de HTTPS. [14]

La siguiente lista muestra los servidores web más populares:

a) Apache

El proyecto Apache HTTP Server es un esfuerzo por desarrollar y mantener un servidor HTTP de código abierto para sistemas operativos modernos, incluidos UNIX y Windows. El objetivo de este proyecto es proporcionar un servidor seguro, eficiente y extensible que proporcione servicios HTTP en sincronía con los estándares HTTP actuales. [15]

El servidor HTTP Apache se lanzó en 1995 y es el servidor web más popular en Internet desde abril de 1996. En febrero de 2020 celebró su 25.º aniversario como proyecto. [15]

El servidor HTTP Apache es un proyecto de **The Apache Software Foundation** . [15]

Características relevantes de este servidor

1. **Función Principal:** Se utiliza para alojar sitios web y servir páginas web a través del protocolo HTTP. Es una parte esencial de la infraestructura de la World Wide Web. [8]
2. **Código Abierto:** Lo que significa que su código fuente está disponible públicamente. Utiliza la Licencia Apache 2.0, esto permite a la comunidad de desarrolladores modificar, mejorar y distribuir Apache de acuerdo con las licencias de software libre, compatible con múltiples plataformas (Windows, Linux, macOS) las cuales son marcas registradas. [15]
3. **Configurabilidad:** Los administradores de sistemas pueden ajustar la configuración para satisfacer las necesidades específicas de un sitio web o una aplicación web. [15]
4. **Extensibilidad:** Apache es altamente extensible a través de módulos. Los módulos son componentes de software que pueden agregarse para proporcionar funcionalidades adicionales, como autenticación, compresión, seguridad y más. [15]
5. **Estabilidad y rendimiento:** Su estabilidad y rendimiento lo hace adecuado para servir sitios web de alto tráfico. Ha sido diseñado para manejar una carga significativa y mantener un funcionamiento robusto. [15]
6. **Seguridad:** Incluye características de seguridad sólidas, lo que lo hace apto para la implementación de conexiones seguras mediante HTTPS. También es compatible con numerosas herramientas de seguridad y autenticación. Es un servidor web altamente personalizable que desempeña un papel en la entrega de contenido web en todo el mundo. [15]

b) Tomcat

El software Apache Tomcat es el motor de numerosas aplicaciones web de gran escala y de misión crítica en una amplia gama de industrias y organizaciones.

Apache Tomcat, Tomcat, Apache, la pluma de Apache y el logotipo del proyecto Apache Tomcat son marcas comerciales de Apache Software Foundation. [16]

Características relevantes de este servidor

- Desarrollado por la Fundación Apache, Tomcat es más bien un contenedor de servlets, pero a menudo se utiliza como servidor web.
- Es ideal para aplicaciones web basadas en Java.

- Permite ejecutar aplicaciones Java EE y se integra bien con servidores web como Apache y Nginx.

c) LiteSpeed

LiteSpeed Web Server es un reemplazo de Apache y puede cargar archivos de configuración de Apache directamente. Se puede utilizar con cualquier panel de control diseñado para Apache, incluidos cPanel, Plesk y DirectAdmin. LiteSpeed no es un proxy de interfaz, sino un reemplazo completo, que incluye motores más eficientes para las funciones populares de Apache, como Rewrite y ModSecurity. [17]

Características relevantes de este servidor

- Un servidor web conocido por su rendimiento y eficiencia en la gestión de tráfico web.
- Es compatible con los archivos de configuración de Apache, lo que facilita la migración.
- Tiene una versión gratuita y una comercial, con características avanzadas en esta última.

1.12 Balanceadores de carga

El balanceador se encarga de reenviar las peticiones a los servidores donde será atendida, actuando como un intermediario entre los clientes y los servidores, cuando recibe una solicitud de un cliente, el balanceador selecciona uno de los servidores disponibles basándose en diferentes criterios, como la cantidad de tráfico, capacidad actual o estado. Una vez seleccionado el servidor, el balanceador reenvía la solicitud para que sea procesada. El objetivo es distribuir las solicitudes de manera equitativa, evitando sobrecargar un solo servidor y garantizando que todos los recursos disponibles sean utilizados de manera eficiente. Estos balanceadores de carga operan de manera tan eficiente que los usuarios no perciben su presencia, ni son conscientes del proceso en el que están participando. [18]

Existen dos tipos de balanceadores que se pueden desarrollar tanto por software, o por hardware, también se pueden combinar, dependiendo la carga de trabajo que se maneje. [19]

a) Balanceo por Hardware

En el balanceo por hardware se utiliza un dispositivo que se encarga de repartir la carga y la envía a los diferentes servidores porque están conectados a este balanceador de carga, aunque estos dispositivos tienen una gran potencia y escalabilidad, estos son muy costosos por el mantenimiento y la configuración de este dispositivo donde el cliente hace una petición y el balanceador la toma directamente, mandará la solicitud dependiendo de cómo esté configurado a uno de los servidores del nodo, el balanceador podrá dirigir el trabajo a otros servidores. [19]

La carga en el balanceador se refiere al volumen de trabajo que un servidor debe procesar en un momento dado. Esta carga incluye el número de solicitudes, la cantidad de conexiones activas, el uso de CPU, memoria, y otros recursos del sistema. [19]

Se considera sobrecargado cuando su capacidad de procesamiento o sus recursos se ven saturados, lo que afecta su rendimiento y capacidad de responder a nuevas solicitudes. Existen varios indicadores para detectar la sobrecarga en un servidor: Esto aplica tanto a balanceo por Hardware como por Software, por lo que la sobrecarga se debe entre otros indicadores a los siguientes:

1. **Alta utilización de memoria:** Si está utilizando un porcentaje muy alto de su memoria, podría estar sobrecargado. Esto indica que el servidor está procesando más solicitudes de las que puede manejar eficientemente.
2. **Aumento en los tiempos de respuesta:** Cuando tarda más de lo habitual en responder a solicitudes, es una señal de que está bajo una carga excesiva.
3. **Conexiones acumuladas:** Si el número de conexiones activas o en cola aumenta significativamente, el servidor podría estar procesando más solicitudes de las que puede manejar.
4. **Errores en las respuestas:** Puede comenzar a generar errores, como respuestas 503 (Error Interno del Servidor) o no poder procesar correctamente las solicitudes.
5. **Latencia en la red:** Un aumento en el tiempo que tarda en transmitirse la información entre el servidor y los clientes también puede ser un indicador de sobrecarga.
6. **Monitoreo de estado:** Se pueden detectar sobrecargas al observar métricas, como uso de recursos, número de conexiones o tiempos de respuesta.

El balanceador de carga utiliza estos indicadores para redistribuir las solicitudes a otros servidores disponibles y evitar la sobrecarga de un solo nodo. En la figura 1.4 se muestra el diagrama de cómo es que funciona este balanceador la sobrecarga se debe otros indicadores a los siguientes. [19]

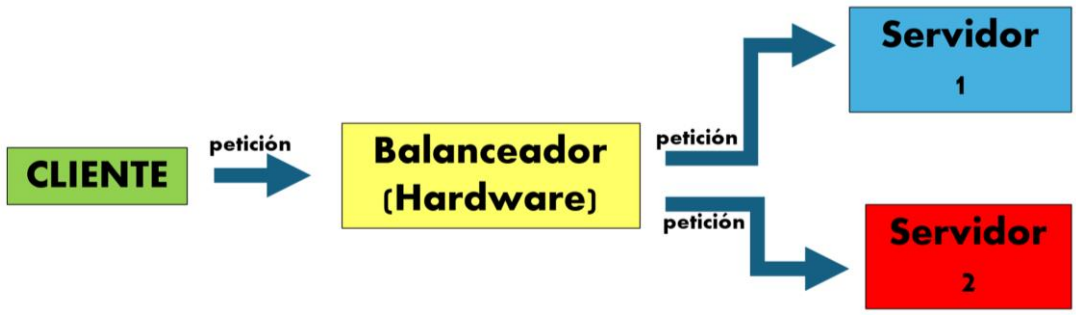

Figura 1.4 Balanceador por hardware. [19]

b) Balanceo por Software

EL balanceo de carga por software es muy parecido al anterior, la diferencia radica en que, en lugar de utilizar un hardware externo, el balanceador se puede desarrollar dentro del mismo sistema operativo de un servidor, mediante una aplicación dentro del sistema. Realiza la misma función, más eficiente y barato que un balanceador de hardware. [19]

Figura 1.5 Funcionamiento de un balanceador mediante software. [19]

En la Figura 1.5, se observa que el cliente realiza la petición y parece que se está conectando a otro servidor. Sin embargo, en realidad, solo se conecta al balanceador (maestro) configurado en esa aplicación, que se encuentra dentro de un sistema operativo. Este balanceador desempeña

la misma función que un balanceador de hardware, distribuyendo la carga según la configuración establecida y dependiendo de la cantidad de carga que se asigna a los demás servidores. [19]

Una vez que el servidor ha procesado una solicitud, el balanceador de carga regresa la respuesta al cliente siguiendo estos pasos:

1. **Procesamiento de la solicitud:** El servidor asignado recibe la solicitud del balanceador y la procesa (por ejemplo, cargando una página web o ejecutando una acción solicitada por el cliente).
2. **Generación de respuesta:** Tras procesar la solicitud, el servidor genera una respuesta, que puede ser algún tipo de servicio.
3. **Envío de respuesta al balanceador:** El servidor devuelve la respuesta al balanceador de carga, quien mantiene el seguimiento de qué solicitud corresponde a qué cliente.
4. **Reenvío al cliente:** El balanceador de carga luego reenvía esta respuesta al cliente que hizo la solicitud inicial. Al cliente le parece que está comunicándose directamente con el servidor, aunque el balanceador gestiona la interacción en segundo plano.

Este proceso es transparente para el cliente, quien recibe la respuesta como si proviniera directamente del servidor original, por lo que aplica para ambos casos, balanceo por Hardware y por Software. [19]

1) Algunos balanceadores

En la siguiente lista se muestran algunos de los balanceadores los cuales se pueden configurar mediante software:

i) Proxy de Alta Disponibilidad (HAProxy)

HAProxy es un proxy inverso (balanceador) gratuito, muy rápido y confiable que ofrece alta disponibilidad , balanceo de carga y proxy para aplicaciones basadas en TCP y HTTP. Es particularmente adecuado para sitios web con mucho tráfico y potencia una parte significativa de los más visitados del mundo. Con el paso de los años se ha convertido en el balanceador de carga de código abierto estándar de facto, ahora se incluye con la mayoría de las distribuciones

de Linux más populares y, a menudo, se implementa de forma predeterminada en plataformas en la nube. Dado que no se anuncia a sí mismo, solo sabemos que se usa cuando los administradores lo informan. [20]

Características de HAProxy

1. **Distribución de la carga:** Distribuye las solicitudes entre múltiples servidores backend[17] para evitar la sobrecarga de uno solo y mejorar la disponibilidad de la aplicación. Esto se hace mediante algoritmos como round-robin [18]o least connections, [19]que asignan solicitudes al servidor más adecuado en función de la cantidad de conexiones activas o la rotación uniforme de solicitudes.

2. **Alta disponibilidad:** Al actuar como un proxy inverso, HAProxy también gestiona la alta disponibilidad. Si un servidor backend falla o se vuelve inestable, el balanceador deja de enviarle solicitudes y redirige el tráfico a otros servidores que estén en buen estado. Esto garantiza que los usuarios no experimenten interrupciones en el servicio.

3. **Seguridad y filtrado de tráfico:** También puede actuar como una primera línea de defensa filtrando solicitudes no válidas y asegurando que solo las solicitudes legítimas lleguen a los servidores backend. Funciona como un firewall[20] de aplicaciones web, protegiendo contra ataques comunes como DDoS[21] o inyecciones de código malicioso.

4. **Escalabilidad:** La versión gratuita, permite a las empresas escalar sus aplicaciones al añadir o quitar servidores backend de manera dinámica, lo que es útil en aplicaciones que experimentan cambios en la demanda de tráfico.

---

[17] La parte trasera (Backend): Es un servidor que gestiona la lógica, el procesamiento de datos y la interacción con base de datos la cual no es visible para el usuario. [20]

[18] Circular (Round robin): Es el método de balanceo de carga que distribuye las solicitudes de manera cíclica entre los servidores disponibles. [20]

[19] Menor cantidad de conexiones (Least Connections) es un método de balanceo de carga que envía solicitud al servidor con menos conexiones activas, buscando distribuir la carga de manera más equilibrada. [20]

[2020] Cortafuegos (Firewall): Es una herramienta de seguridad de red que supervisa y controla el tráfico de datos entre redes, permitiendo o bloqueando el acceso según las reglas de seguridad predefinidas. [20]

[21] Ataque Distribuido de Denegación de Servidio (DDoS, por sus siglas en inglés: Distributed Denial lof Service): Es un tipo de ataque cibernético en el que se envían grandes cantidades de tráfico o solicitudes falsas de un servidor, red o sitio web. [20]

5. **Monitoreo y salud de los servidores:** HAProxy incluye funciones para realizar chequeos de salud en los servidores backend. Esto permite saber si un servidor está disponible antes de enviarle tráfico. Si un servidor falla en estos chequeos, HAProxy lo excluye del grupo de servidores hasta que vuelva a estar disponible.

Esta solución es gratuita, adecuada para pequeñas y grandes infraestructuras, y es usada ampliamente por empresas de todo el mundo para manejar grandes volúmenes de tráfico web sin costo de licencias [20]

ii) Apache HTTP Server con mod_proxy

mod_proxy y los módulos relacionados implementan un proxy/puerta de enlace para Apache HTTP Server, que admite una serie de protocolos populares como: HTTP, HTTPS, FTP, así como varios algoritmos de equilibrio de carga diferentes. Los módulos de terceros pueden agregar compatibilidad con protocolos adicionales y algoritmos de equilibrio de carga. [21]

Se debe cargar un conjunto de módulos en el servidor para proporcionar las funciones necesarias. Estos módulos se pueden incluir de forma estática en el momento de la compilación o de forma dinámica mediante la directiva **LoadModule**. El conjunto debe incluir: [21]

- **mod_proxy**, que proporciona capacidades de proxy básicas
- **mod_proxy_balancer** y uno o más módulos balanceadores si se requiere balanceo de carga.

1.13 Planteamiento del Problema

En la actualidad, el uso masivo de páginas web ha llevado a un incremento en las solicitudes de acceso a servidores web, lo cual puede resultar en la saturación de los recursos de un servidor individual. Este fenómeno, conocido como Denegación de Servicio (DoS), ocurre cuando un servidor es sobrecargado con más peticiones de las que puede manejar, afectando negativamente su capacidad de procesamiento, acceso a disco, y el manejo de tráfico de red, lo que eventualmente lo deja fuera de servicio.

El DoS es un problema para la disponibilidad y el rendimiento de los servicios web en las operaciones diarias de las organizaciones e instituciones. Para moderar este problema, se ha implementado una solución que consiste en distribuir la carga de trabajo entre múltiples servidores, lo que se conoce como balanceo de carga. Esta técnica permite que cada servidor maneje solo una parte de las solicitudes, reduciendo así el riesgo de saturación.

No obstante, para que los usuarios experimenten un acceso fluido al sitio web, independientemente de la distribución del servicio entre varios servidores, el sistema debe presentar una única dirección IP o nombre de dominio. Esto se logra mediante la implementación de un balanceador de carga, que dirige de manera eficiente las peticiones a los servidores correspondientes, garantizando la continuidad y calidad del servicio.

El problema que aborda este trabajo es la necesidad de caracterizar el funcionamiento de un sistema de balanceo de carga en términos de su consumo de recursos en un entorno controlado de servidores distribuidos dentro de una red de área local (LAN). Esto es esencial para entender cómo optimizar la utilización de los recursos en un entorno distribuido y mejorar la solidez y disponibilidad del servicio web.

1.14 Justificación

El problema que aborda este trabajo es la necesidad de caracterizar el funcionamiento de un sistema de balanceo de carga, específicamente utilizando HAProxy, en un entorno controlado de servidores distribuidos dentro de una red de área local (LAN). Entender y caracterizar el sistema de balanceo es esencial para optimizar los recursos del servidor y mejorar el servicio, reducir fallas, así como ampliar la disponibilidad del servicio, entre otros. En este contexto, es evaluar eficientemente las solicitudes entre los servidores disponibles, y evaluar ante la carga de tráfico cómo es su comportamiento en cuestión de recursos, tanto del servidor como del balanceador. Para generar tráfico se necesita una o varias herramientas, esto también es crucial porque con ello se causa estrés al servidor.

Utilizando Siege, una herramienta de pruebas de carga se ha simulado un entorno con múltiples instancias y usuarios concurrentes. Esto permite analizar el comportamiento del balanceador de carga y su capacidad para manejar el tráfico de forma eficiente. En particular, se ha empleado HAProxy para distribuir las solicitudes de manera equilibrada, utilizando métodos como Round-

robin y Least Connections, que optimizan la asignación de carga según la cantidad de tráfico y el número de conexiones activas en cada servidor.

El uso de Round-robin, que distribuye las solicitudes de manera cíclica entre los servidores, permite evaluar cómo el balanceador distribuye el tráfico de forma uniforme, mientras que Least Connections, que prioriza los servidores con menos conexiones activas, asegura que las cargas se gestionen de manera eficiente, minimizando el riesgo de sobrecarga en los servidores. Estos métodos, al ser probados en un entorno controlado, proporcionan información sobre cómo los balanceadores de carga contribuyen a la optimización de los recursos.

El análisis de los resultados, utilizando herramientas como Siege, ayuda a comprender cómo el balanceador de carga afecta el rendimiento del servidor y la capacidad de manejar grandes volúmenes de tráfico. De esta forma, es posible evaluar el consumo de recursos en un entorno distribuido y, al mismo tiempo, mejorar la disponibilidad y la solidez del servicio web, lo cual es esencial para garantizar una experiencia de usuario eficiente y continua.

1.15 Metodología

Objetivo General

Conocer un sistema Balanceador de Carga, su configuración y características en un ambiente local controlado para indicar las ventajas con respecto de su utilización para expandir las capacidades de atención a usuarios de un sitio web.

Objetivos Específicos

- Comprender el funcionamiento y las características de un balanceador de carga.
- Configurar un entorno de servidores distribuidos en una LAN.
- Evaluar el desempeño del balanceador de carga en diferentes condiciones de tráfico.

1. **Diseño del sistema.**

- **Seleccionar una topología de red:** Diseñar la topología que se utilizara en la configuración adecuada de los servidores web, balanceador de carga, incluir otros componentes de la red como routers o switches.

- **Selección de herramientas**: Definir los sistemas operativos y software de servidor web, seleccionar software de balanceo de carga y determinar herramientas de monitoreo y generación de tráfico.

2. Instalación y configuración.

- **Preparación del entorno:** Configurar el hardware y la infraestructura necesaria, incluyendo la instalación de servidores físicos o virtuales.
- **Instalación del Balanceador de Carga:** Instalar el software seleccionado para el balanceo de carga.
- **Configuración Inicial:** Configurar el balanceador de carga para distribuir el tráfico entre los servidores según la estrategia de balanceo seleccionada.
- **Configuración de Servidores Web:** Configurar los servidores web en la red LAN para recibir peticiones distribuidas por el balanceador.
- **Configuración de Monitoreo:** Configurar herramientas de monitoreo para supervisar el tráfico de red, el uso de recursos del balanceador de carga.

3. Pruebas y Evaluación.

- **Pruebas de Carga:** Realizar pruebas de carga para simular diferentes niveles de tráfico y evaluar el desempeño del balanceador de carga,
- **Medición del Desempeño:** Se realiza con las tres mediciones siguientes: Registrar el tiempo de respuesta, la distribución de tráfico entre servidores, y el consumo de recursos.
- **Análisis de Resultados:** Analizar los resultados de las pruebas para identificar cuellos de botella, eficiencia del balanceo.

4. Conclusiones.

- Mostrar los resultados, destacando como funcionó el balanceador de carga, como ayudó a qué los servidores web fueran más resistentes[22] y sugerir mejoras para futuras implementaciones.

[22] Al mencionar un servidor más resistente, quiere decir que este soporta más tráfico de red si es lanzado a este mismo.

REFERENCIAS CAPÍTULO 1

[1] A. S. T. y. D. J. Wetherall, Redes de computadoras, México: Pearson Educación de México, S.A de C.V, 2011.

[2] W. Stalling, Comunicaciones y Redes de Computadores, Madrid : Pearson, 2004

[3] S. F. M. P. J. García, «Redes para Proceso Distribuido,» de *Redes para Proceso Distribuido*, Madrid, RA-MA, 2000, p. 750.

[4] KIO, «¿Qué son los dispositivos de interconexión de redes y cuáles son los mejores?,» One Step Forward, 2019. [En línea]. Available: https://www.kio.tech/blog/data-center/dispositivos-de-interconexion-de-redes#:~:text=Los%20dispositivos%20clave%20que%20comprenden,Una%20sola%20LAN. [Último acceso: 10 agosto 2024].

[5] A. Estrada, «Protocolo TCP/IP de Internet,» *Revista Digital Universitaria,* p. 7, 10 septiembre 2004.

[6] D. P. Torrez, Redes CISCO Curso práctico de formación para la certificación CCNA, México: Alfaomega, 2018.

[7] C. d. U. A. y. E. a. Distancia, «CUAED UNAM,» 2017. [En línea]. Available: https://repositorio-uapa.cuaieed.unam.mx/repositorio/moodle/pluginfile.php/2783/mod_resource/content/1/UAPA-Fundamentos-Redes-Computadoras/index.html. [Último acceso: 20 agosto 2024].

[8] S. Cadavilla, «Explicando BSD,» FreeBSD, 22 noviembre 2021. [En línea]. Available: https://docs.freebsd.org/es/articles/explaining-bsd/#:~:text=Es%20el%20nombre%20de%20las,conocida%20como%204.4BSD%2DLite. [Último acceso: 22 agosto 2024].

[9] FYCGROUP, «UNIX: La simplicidad del ingenio,» 10 noviembre 2020. [En línea]. Available: https://www.fyccorp.com/articulo-unix:-la-simplicidad-del-ingenio#:~:text=El%20nombre%20UNIX%20fue%20inicialmente,cumplan%20con%20la%20%E2%80%9CSingle%20Unix. [Último acceso: 23 agosto 2024].

[10] I. E. T. F. (IETF), «Protocolo de transferencia de hipertexto versión 2 (HTTP/2) RFC 7540,» IETF, 21 enero 2020. [En línea]. Available: https://datatracker.ietf.org/doc/rfc7540/. [Último acceso: 23 agosto 2024].

[11] IBM, «Método GET, POST para crear un recurso de actividad nuevo,» 25 enero 2024. [En línea]. Available: https://www.ibm.com/docs/es/baw/22.x?topic=cnar-get-method. [Último acceso: 7 noviembre 2024].

[12] D. C. Herrera, «URI Y URL: Diferencias y cuando usarlas,» HASTINGER, 6 septiembre 2024. [En línea]. Available: https://www.hostinger.mx/tutoriales/uri-vs-url#:~:text=URL%20es%20el%20subconjunto%20de,la%20ruta%20y%20la%20query.. [Último acceso: 9 noviembre 2024]

[13] InternationalWeb, «Noticias & Novedades. Blog qué es la Web 2.0,» 12 abril 2008. [En línea]. Available: https://www.internacionalweb.com/noticias/que-es-la-web-2.0.htm. [Último acceso: 22 agosto 2024].

[14] S. Casas, Servidor Web Apache 2.4 (LINUX), España: International CC, 2021.

[15] A. H. S. PROJECT, «Documentación,» 2024. [En línea]. Available: https://httpd.apache.org/. [Último acceso: 5 septiembre 2024].

[16] T. A. S. FOUNDATION, «Apache Tomcat,» Apache, 2024. [En línea]. Available: https://tomcat.apache.org/. [Último acceso: 6 septiembre 2024].

[17] LITESPEED, «Servidor Web LiteSpeed,» LITESPEED, 2024. [En línea]. Available: https://www.litespeedtech.com/products/litespeed-web-server. [Último acceso: 6 septiembre 2024].

[18] aws, «¿Qué es el equilibrio de carga?,» Amazon, 2023. [En línea]. Available: https://aws.amazon.com/es/what-is/load-balancing/. [Último acceso: 28 agosto 2024].

[19] R. J. Edwars, Implementación de un balanceador de carga y alta disponibilidad para servicios web, México: UNAM, 2015.

[20] HAPROXY, «The Reliable, High Performance TCP/HTTP Load Balancer,» 2024. [En línea]. Available: https://www.haproxy.org/. [Último acceso: 7 septiembre 2024]

[21] A. H. S. PROJECT, «Modulo Apache mod_proxy,» 2024. [En línea]. Available: https://httpd.apache.org/docs/current/mod/mod_proxy.html. [Último acceso: 7 septiembre 2024].

CAPÍTULO 2

CONFIGURACIÓN DEL AMBIENTE LOCAL CONTROLADO

2.1 Configuración de red de área local (LAN)

Actualmente, la demanda en servicios web dinámicos y eficientes es mayor que nunca. Las organizaciones dependen cada vez más de sus servidores web para proporcionar acceso continuo a aplicaciones y contenidos en línea, sin embargo, esto requiere de mucha carga de trafico de red, un solo servidor a veces puede verse sobrepasado y puede llevar a sobrecargarse y por ende a tiempos de respuesta lentos e incluso fallas en el sistema. Para atender este problema, es necesario aplicar la distribución de carga de trabajo, entre varios servidores.

Un balanceador de carga actúa como un intermediario entre los usuarios y los servidores, distribuyendo las peticiones de manera equitativa, mejora la eficiencia y una mejor manera de manejar el tráfico de red y fenómenos de denegación de servicio (DoS).

Esta introducción establece el contexto para el desarrollo de una red LAN que se enfocará para desarrollar el balanceador de carga, explicando la importancia, el rendimiento y la disponibilidad de los servicios web. A continuación, en la figura 2.1 se muestra la estructura de la topología propuesta para el desarrollo del proyecto propuesto.

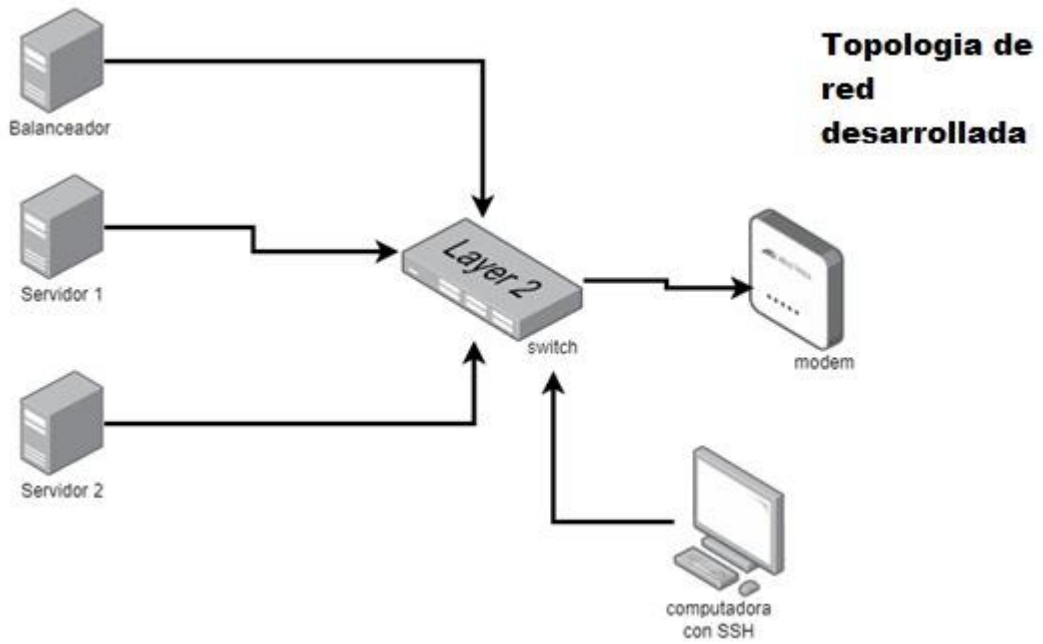

Figura 2.1 Topología de la red propuesta [Autoría propia].

La red LAN propuesta contará con tres servidores, dos de los cuales atienden las peticiones y el tercero fungirá como balanceador, una computadora con conexión SSH[23] para poder configurar el balanceador y los servidores, después se agregará otra máquina la cual será la responsable de generar tráfico con el fin de poner a prueba el balanceador de carga.

a) Direccionamiento IP, Router Residencial

Para el desarrollo de la red se necesita asignar una IP única a cada dispositivo para que se puedan comunicar entre sí dentro de la red, las direcciones IP actúan para identificar a qué dispositivo se le envía y reciban la información de manera efectiva.

Para desarrollar esta red se usará un router residencial el cual será la base, se iniciará conectando el modem al router residencial para poder tener conexión a internet, el cual se encargará de dar el direccionamiento al modem residencial, está configurado con el protocolo DHCP (Dynamic Host Configuration Protocol) el cual se encarga de tener IP dinámicas, esto implica que la dirección IP asignada a un dispositivo cambie después de un período de tiempo determinado de

[23] Protocolo Seguro de Acceso Remoto (SSH, por sus siglas en inglés: Secure Shell): Es un protocolo de red seguro que permite la conexión remota entre equipos mediante una red, generalmente utilizada para administrar servidores de forma segura. SSH encripta todos los datos transmitidos. [1]

manera automática. Sin embargo, cuando se configura un balanceador de carga, las IP dinámicas no son adecuadas, ya que el constante cambio en las direcciones dificulta que el balanceador sepa a qué servidor dirigir el tráfico de red, esto afecta la eficiencia del sistema, dado que la IP de los servidores se actualiza constantemente, y el balanceador no puede localizar de manera fiable el destino de las solicitudes. La solución a este problema es utilizar IP estáticas, lo que permite que el balanceador de carga identifique de manera permanente y sin inconvenientes a cada servidor, garantizando que las solicitudes se distribuyan correctamente.

Por lo que se usará la función de fijar las IP. A continuación, en la tabla 2.1 se muestra el direccionamiento y las IP fijas que se asignaron a cada dispositivo que fungirán como servidores y el balanceador.

Descripción	IP asignada
Manager (Balanceador)	192.168.1.67
Nodo 1	192.168.1.64
Nodo 2	192.168.1.69

Tabla 2.1 Direccionamiento IP de los nodos y el balanceador.

b) Colocación de los dispositivos

Como se muestra en la figura 2.1, la configuración incluye un balanceador y varios servidores conectados a un switch. Este switch se encargará de distribuir la conexión a través de toda la red. Esta configuración se implementó de esta manera para separar físicamente los dispositivos, para prevenir interferencias con la red local doméstica.

c) Dispositivos de interconexión

A continuación, se muestran las especificaciones de cada dispositivo.

SWITCH TP-LINK TL_SF1008D DE 8 PUERTOS

Características descritas por el fabricante:

- 8 puertos RJ4510/100Mbps

- Tecnología Green Ethernet ahorra la energía hasta un 70%. Es un conjunto de técnicas diseñadas para reducir el consumo de energía en redes Ethernet, especialmente en dispositivos como switches y routers. Fue desarrollada como respuesta al creciente uso de redes y la necesidad de mejorar la eficiencia energética.
- Protocolo IEEE 802.3x control de flujo proporciona una transferencia de datos fiables

Los 8 puertos del switch proporcionan espacio adicional para añadir más nodos si así se desea. [2]

Principales características de Green Ethernet:

1. **Detección de Longitud de Cable:** Esta tecnología ajusta la potencia de transmisión según la longitud del cable Ethernet. Si el cable es más corto, se requiere menos potencia para transmitir datos, lo que permite ahorrar energía.
2. **Estado de Bajo Consumo (Idle):** Cuando un puerto Ethernet no está en uso, Green Ethernet reduce la energía consumida por dicho puerto o lo pone en modo de bajo consumo. Esto es útil para redes con muchos dispositivos conectados pero que no siempre están activos.
3. **Auto-Negociación de Potencia:** Green Ethernet puede ajustar el consumo energético de los dispositivos en función de la cantidad de tráfico que pasa por ellos. Si hay menos tráfico, se reduce el consumo de energía.

La tecnología Green Ethernet permite que las redes Ethernet sean más eficientes energéticamente, reduciendo el uso innecesario de energía y haciendo que las redes sean más cuidadosas con el medio ambiente. [3]

2.2 Dispositivos de la red propuesta

a) Raspberry Pi

“Una computadora de placa única de bajo costo, también conocida como "Single Board Computer" (SBC, por sus siglas en inglés), es una computadora de tamaño compacto, similar al de una tarjeta de crédito, Esta placa está equipada con un procesador, memoria y puertos de entrada y salida, puede conectarse a un monitor, a una televisión con entrada HDMI, y utilizarse con un mouse y teclado estándar, así como con una cámara de video y audio. Sus características

la hacen ideal para desarrollar prototipos, modelos educativos, aplicaciones de domótica o como controlador en dispositivos del “Internet de las cosas” (IoT)”. [4]

b) Resumen de características de la Raspberry Pi

La Raspberry Pi 4 está equipada con las siguientes características:

- **CPU (Central Processing Unit):** Es el componente central de procesamiento que se encarga de ejecutar las instrucciones y realizar las operaciones básicas necesarias para que el sistema funcione. Es el "cerebro" de la Raspberry Pi, gestionando todo el procesamiento de datos y controlando los demás componentes del sistema.

Características del CPU.

1. **Arquitectura ARM Cortex-A72**

   Es una arquitectura de 64 bits diseñada para ofrecer un alto rendimiento y eficiencia de energía, lo que es ideal para este dispositivo.

2. **Cuatro Núcleos**

   El CPU tiene cuatro núcleos de procesamiento, lo que permite ejecutar múltiples tareas simultáneamente (multitarea) de manera eficiente. Esto mejora significativamente el rendimiento en comparación con las versiones anteriores.

3. **Velocidad de Reloj de 1.5 GHz**

   La velocidad de reloj de 1.5 GHz permite un procesamiento rápido en ejecución de instrucciones, multitareas, esto permite ejecutar varias tareas en paralelo mejorando la eficiencia, lo que hace que sea adecuada para aplicaciones que requieren procesamientos simultáneos, como la navegación web, reproducción de video en alta definición, servidores web.

4. **Procesador Quad-Core de 64 bits**

   Esto permite manejar más memoria y realizar operaciones más complejas, lo que mejora el rendimiento general del sistema, especialmente en aplicaciones que requieren un procesamiento intensivo.

- **GPU (Graphics Processing Unit):** Es el responsable del procesamiento de gráficos y multimedia. Esto es especialmente útil para aplicaciones que requieren renderización de video o gráficos.

- **Memoria LPDDR4**: Es donde se almacenan los datos y las instrucciones que están siendo utilizadas activamente por el sistema operativo y los programas en ejecución.

1. **Low-Power (LP)**: Indica que la memoria está optimizada para consumir menos energía que las versiones estándar de DRAM, lo que es crucial para dispositivos móviles y portátiles.
2. **Double Data Rate (DDR)**: Significa que la memoria puede transferir datos en ambos lados de subida y bajada del ciclo del reloj, lo que duplica la tasa de transferencia de datos en comparación con la SDRAM convencional.

- **Puertos USB (Universal Serial Bus):** La Raspberry está equipada con dos versiones de puertos USB:

1. **USB 3.0**

Cantidad: 2 puertos.

Velocidad de Transferencia: Hasta 5 Gbps (gigabits por segundo)

2. **USB 2.0**

Cantidad: 2 puertos.

Velocidad de Transferencia: Hasta 480 Mbps (megabits por segundo)

Estos puertos permiten la conexión de una amplia variedad de dispositivos y periféricos, desde unidades de almacenamiento externas hasta teclados y ratones, aprovechando las capacidades de alta velocidad del USB 3.0 para tareas que requieren mayor ancho de banda.

- **Puertos HDMI (High Definition Multimedia Interface):** Tiene dos puertos HDMI que permiten conectarla a monitores o televisores para la salida de video.
- **Puerto Ethernet:** Es una interfaz física que permite la conexión de dispositivos de red para la transmisión de datos a través de cables Ethernet. Este puerto es comúnmente utilizado para conectar computadoras, routers, switches, y otros dispositivos de red en una red de

área local (LAN), permitiendo la comunicación y el intercambio de datos entre ellos. Estos puertos utilizan conectores RJ-45.

Funciones principales del puerto Ethernet:

1. **Transmisión de datos:** Permite la transferencia de datos entre dispositivos, como computadoras y servidores, a través de cables de red.

2. **Conexión estable y rápida:** A diferencia de las conexiones inalámbricas, los puertos Ethernet ofrecen una conexión más rápida y estable.

3. **Compatibilidad con estándares de red:** Son compatibles con diferentes velocidades de transmisión, como Fast Ethernet (100 Mbps), Gigabit Ethernet (1000 Mbps) e incluso 10 Gigabit Ethernet (10 Gbps).

- **Wi-Fi y Bluetooth:** Tiene soporte integrado para Wi-Fi y Bluetooth, lo que permite la conectividad inalámbrica a redes locales y dispositivos periféricos compatibles.
- **Puertos GPIO (General Purpose Input/Output):** Estos puertos permiten la conexión de componentes electrónicos y periféricos externos, lo que brinda flexibilidad para proyectos de hardware.
- **Ranura para tarjeta microSD:** La tarjeta microSD se utiliza para almacenar el sistema operativo y los datos del usuario.
- **Puerto de alimentación:** Este es el puerto donde se conecta el adaptador de corriente para suministrar energía de la placa. [4], [5]

2.3 Sistema operativo seleccionado

a) Distribución Debian

Debian es una de las distribuciones así denominados los sistemas operativos de Linux antiguas y respetada, conocida por su estabilidad, seguridad, y software libre. Incluye el núcleo de Linux (kernel), una colección de software, herramientas de administración, y un sistema de gestión de paquetes que facilita la instalación y actualización del software. Fue fundada en 1993 por Ian

Murdock y ha crecido hasta convertirse en una de las bases más importantes para muchas otras distribuciones.

1. **Estabilidad:** Las versiones estables pasan por un proceso riguroso de pruebas antes de ser lanzadas. Esto la hace ideal para servidores, donde la confiabilidad es crucial. La instalación puede configurarse fácilmente para cumplir diversas funciones, desde cortafuegos[24] reducidos al mínimo, a estaciones de trabajo científicas o servidores de red de alto rendimiento.

2. **Seguridad:** Las actualizaciones de seguridad se gestionan de manera centralizada y se distribuyen rápidamente para mantener los sistemas seguros. Para proteger su sistema contra "caballos de Troya" y otros programas malévolos, los servidores de Debian verifican que los paquetes provienen de los desarrolladores oficiales. Se proporcionan actualizaciones y correcciones muy rápidamente si se descubren problemas de seguridad en los paquetes ya distribuidos. [7]

3. **Comunidad Activa:** Cuenta con una gran comunidad de desarrolladores y usuarios que contribuyen al mantenimiento, desarrollo, y soporte de la distribución. Esto asegura que Debian esté siempre actualizado y respaldado.

4. **Paquetes y Repositorios:** Ofrece una gran colección de software a través de sus repositorios, que incluyen miles de paquetes recompilados y listos para instalar, facilitando a los usuarios la instalación y gestión del software.

Muchas distribuciones derivadas, como Ubuntu, Linux Mint, y Raspbian (ahora Raspberry Pi OS), utilizan Debian como base, de ahí su importancia en Linux. [7]

b) Antecedentes Raspberry Pi OS

El sistema operativo original, Raspbian, fue desarrollado por Peter Green y Mike Thompson para aprovechar al máximo el hardware limitado de la Raspberry Pi. Se basa en Debian, pero está optimizado específicamente para el procesador ARM de las Raspberry Pi. Con el tiempo,

[24] Un cortafuegos (o firewall): Es un sistema de seguridad que controla y filtra el tráfico de red entre diferentes áreas de una red informática, como entre una red interna y el internet. Su objetivo principal es bloquear accesos no autorizados y permitir solo el tráfico seguro y autorizado, protegiendo así los recursos de la red. [6]

Raspbian fue ganando popularidad gracias a su ligereza y al creciente interés en la Raspberry Pi como una herramienta educativa y de desarrollo. En mayo de 2020, la Fundación Raspberry Pi decidió renombrar Raspbian como Raspberry Pi OS para reflejar su estatus como el sistema operativo oficial de la Raspberry Pi. La evolución del sistema operativo es el éxito de la plataforma Raspberry Pi, que sigue siendo una de las herramientas educativas y de desarrollo más accesibles y versátiles disponibles en el mercado. [8]

c) Sistema Operativo Raspberry Pi Lite

Raspberry Pi Lite es una versión más ligera del sistema operativo oficial Raspberry Pi OS, diseñada para funcionar en la placa Raspberry Pi con un uso muy bajo de recursos. Esta versión no incluye un entorno de escritorio, lo que la hace ideal para proyectos que requieren un sistema operativo ligero y eficiente. [9]

Características

1. **Sin Entorno Gráfico:** No incluye un entorno de escritorio. Se maneja exclusivamente desde la terminal del sistema, lo que reduce el consumo de recursos como la memoria y el almacenamiento.
2. **Proyectos Específicos:** Está optimizado para proyectos que no necesitan una interfaz gráfica, como servidores, proyectos de automatización, dispositivos IoT. Ideal para usuarios avanzados que prefieren configurar y manejar sus proyectos desde la terminal.
3. **Rendimiento y Velocidad:** Al no tener que cargar componentes gráficos, el sistema operativo se inicia más rápido y utiliza menos recursos, lo que mejora el rendimiento general en dispositivos con hardware limitado, como la Raspberry Pi Zero o versiones más antiguas de la Raspberry Pi.

Características	Raspberry Pi Zero (1.3 / W)	Raspberry Pi 4 (Model B)
Procesador	ARM11 1 GHz (Single-core)	Cortex-A72 1.5 GHz (Quad-core)
Memoria RAM	512 MB	2 GB, 4 GB o 8 GB LPDDR4
Puertos USB	1 Micro-USB OTG	2 USB 3.0, 2 USB 2.0
Salida de video	Mini-HDMI (1080p)	2 Micro-HDMI (4K)
Ethernet	No	Gigabit Ethernet
Wi-Fi y Bluetooth	Solo en Raspberry Pi Zero W	Wi-Fi 802.11ac, Bluetooth 5.0
Almacenamiento	MicroSD	MicroSD
Puertos GPIO	40 pines (sin soldar en Zero)	40 pines
Tamaño	Compacto (65mm x 30mm)	Más grande (85.6mm x 56.5mm)

Tabla 2.2 Tabla comparativa entre la **Raspberry Pi Zero** y la **Raspberry Pi 4**, destacando sus diferencias en términos de hardware. [4], [9]

4. **Compatibilidad:** Es compatible con la mayoría de las aplicaciones y herramientas disponibles para Raspberry Pi OS. Los usuarios pueden instalar y configurar los servicios y paquetes que necesitan a través de la terminal.

5. **Tamaño de la Imagen:** El tamaño de la imagen del sistema operativo es de 432 MB más pequeña en comparación con la versión completa, lo que facilita su instalación en tarjetas SD de menor capacidad y reduce el tiempo de descarga y configuración.

Es comúnmente utilizado para servidores web, servidores de archivos, sistemas de control e instalando solo los componentes y servicios necesarios para su aplicación específica. [9]

d) Kernel Linux 6.6 LTS

El Kernel es la interfaz entre el hardware de una computadora y sus procesos. Se le llama kernel porque se encuentra como núcleo del Sistema Operativo controlando todas las funciones principales del hardware.

El kernel Linux 6.6 LTS (Long-Term Support) es una versión del núcleo de Linux que ha sido designada para recibir soporte a largo plazo, liberado el 29 de octubre de 2023, siendo diciembre de 2026 el fin de vida útil proyectado. [10]

Características de Linux 6.6 LTS

1. **Estabilidad y Soporte Prolongado:** Está diseñado para ser una base estable para sistemas operativos, servidores, y dispositivos embebidos.

2. **Mejoras y Nuevas Funcionalidades:** Incluye mejoras en el rendimiento, soporte para nuevo hardware, optimizaciones en la gestión de recursos, y nuevas funcionalidades. Puede incluir soporte para nuevos procesadores, optimización de controladores de hardware, y mejoras en la seguridad.
3. **Seguridad: R**ecibe actualizaciones de seguridad para asegurar que los sistemas que lo utilizan estén protegidos contra vulnerabilidades conocidas.
4. **Compatibilidad:** Se asegura la compatibilidad con una amplia gama de hardware, desde servidores de alto rendimiento hasta dispositivos embebidos y sistemas de computación en la nube. [10], [11], [12]

e) Forma de instalación

Paso 1

Se necesita descargar y guardar en una memoria Micro SD el software desde el siguiente enlace https://www.raspberrypi.com/software/ en esta página se puede descargar el creador de imágenes de Raspberry Pi oficial, en la Figura 2.2 muestra la interfaz de este software la cual es una herramienta esencial para poder llevar a cabo la instalación del sistema operativo.

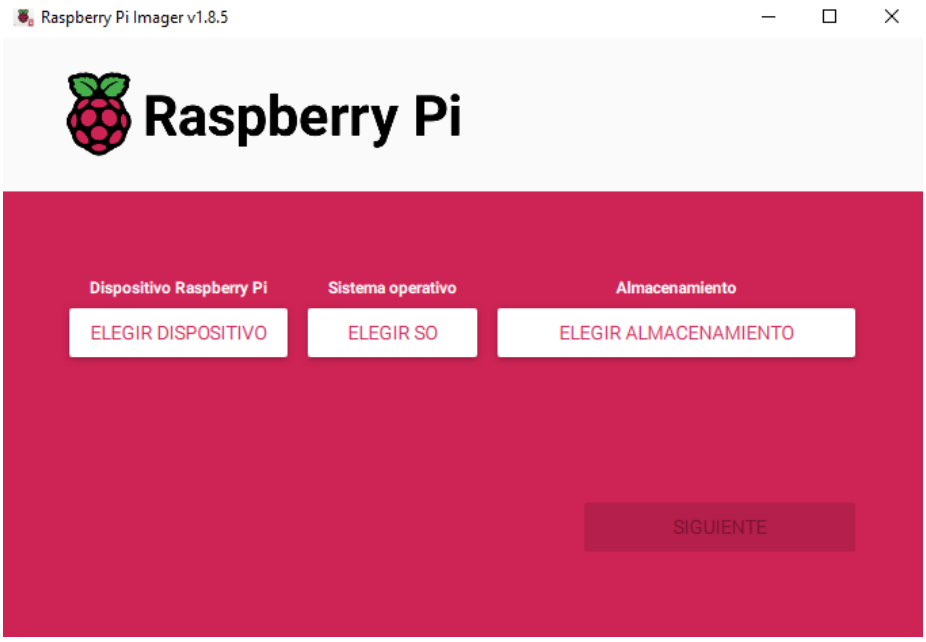

Figura 2.2. Interfaz gráfica software oficial de Raspberry Pi para crear imágenes de sistemas operativos. [4]

Paso 2

Una vez teniendo instalado este software como se puede ver en la Figura 2.2 en el apartado de **ELEGIR DISPOSITIVO** se elige el modelo del cual se tenga disponible.

Paso 3

En el apartado de **ELEGIR SO** se elige el sistema el cual se vaya a utilizar en los cuales tiene varias distribuciones como Raspberry Pi OS, UBUNTU, Debian, entre otros, ya sea la elección de 32 bits y 64 bits esto dependiendo del tipo de procesador que se tenga en el dispositivo.

Paso 4

Después se deberá **ELEGIR ALMACENAMIENTO** se elige la memoria en la cual se quiera grabar el sistema operativo, se recomienda que la memoria sea al menos de 16 GB y se tenga un adaptador de USB a MicroSD para que sea más rápido el procedimiento.

Paso 5

Una vez teniendo estos parámetros se hace clic en el botón **SIGUIENTE** de ahí, antes de escribir la imagen se necesitan seleccionar algunas características, a grandes rasgos ésta es la forma en que se instala el sistema operativo dependiendo de la necesidad del proyecto que se esté desarrollando

f) Características instaladas

Una de las características más importantes es el poder activar el protocolo SSH, el cual es necesario si se quiere manejar de manera remota y con los menores recursos posibles, para poder llevar a cabo esta característica se necesita hacer el siguiente procedimiento.

Paso 1

Una vez que se selecciona el botón **SIGUIENTE** desplegará una ventana como se muestra en la figura 2.3, pregunta si se quiere personalizar el SO.

Figura 2.3 Personalización del S.O [4]

Paso 2

Una vez que muestra esta ventana se debe seleccionar **EDITAR AJUSTES** en donde se selecciona esa característica de personalizar el nombre de usuario con su respectiva contraseña, también se puede editar el hostname(anfitrión) y así poder identificar los dispositivos en la red, como se muestra en la figura 2.4.

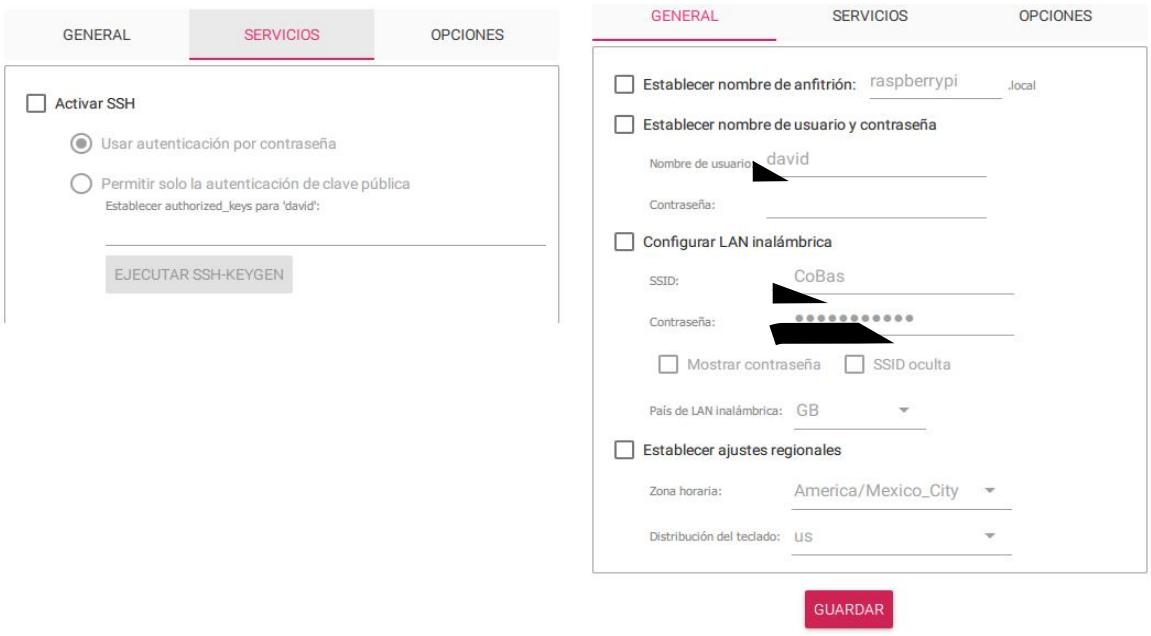

Figura 2.4 Selección de la característica para poder activar el protocolo SSH.

Paso 3

Como se puede observar en la imagen 2.4 tiene la opción de activar SSH directamente al grabar la imagen, también se puede definir el usuario y contraseña, esta es la característica más importante para que podamos configurarlo de manera remota.

2.4 Servidores instalados

Se instalaron dos servidores backend. Se instaló Apache lo que permitió configurar un servidor web robusto y flexible que maneja las solicitudes HTTP. Ambos servidores se configuraron adecuadamente para trabajar en conjunto con HAProxy, permitiendo una distribución eficaz del tráfico y asegurando la disponibilidad continua de los servicios.

a) Configuración máxima de peticiones en Apache

Workers se refiere a los procesos o hilos que manejan las solicitudes en un servidor web. Su configuración adecuada es crucial para el rendimiento y la estabilidad del servidor. Como lo menciona en la documentación de Apache "**La Directiva MaxRequestWorkers.** establece el límite de la cantidad de solicitudes simultáneas que se atenderán, cualquier intento de conexión que supere el límite se pondrá en cola, hasta un número determinado según **La Directiva Listen Backlog**. Una vez que se libera un proceso secundario al final de una solicitud diferente, se atenderá la conexión." [13]

b) Número de procesos lanzados

"En el servidor, existe un límite estricto de ServerLimit de 20,000 archivos compilados, mientras que para **MPM Prefork** [25]este límite se eleva a **200,000**. Esta restricción tiene como propósito evitar problemas indeseados que puedan surgir por errores tipográficos. Si se desea aumentar este límite más allá de lo establecido, será necesario modificar el valor de MAX_SERVER_LIMIT en el archivo fuente de MPM y luego reconstruir el servidor." [13] Esto

[25] Módulo de multiprocesamiento Prefork (MPM Prefork, por sus siglas en inglés: Multi-Processing Module Prefork): Es un módulo de Apache que gestiona la forma en que el servidor maneja las solicitudes entrantes mediante procesos en lugar de hilos. Es ideal para aplicaciones que no son "thread-safe" (no seguras para trabajar con hilos) porque mantiene cada solicitud en su propio proceso independiente, evitando la posibilidad de conflictos entre solicitudes. [14]

quiere decir que establece un límite superior en el número de procesos de trabajo (worker) que Apache puede crear. Esto es útil para evitar que el servidor consuma demasiados recursos (como memoria y CPU) en situaciones de alta demanda. El valor de **ServerLimit** debe ser igual o mayor que **MaxRequestWorkers**. Este define el número máximo de peticiones concurrentes que Apache puede manejar. Si se desea aumentar **MaxRequestWorkers** por encima del valor por defecto, también se debe aumentar **ServerLimit**. [13]

2.5 Razones para la selección del balanceador HAproxy

La elección de este sistema se basa en su configuración, que es relativamente sencilla en comparación con otros balanceadores de carga. A diferencia de soluciones como **NGINX**[26], que puede actuar tanto como servidor web y balanceador de carga, HAProxy está diseñado específicamente para la función de balanceo, lo que optimiza su rendimiento en esta tarea. Otra razón significativa es su alta disponibilidad, distribuye eficientemente el tráfico, incluso bajo cargas pesadas, asegurando que el sistema mantenga su rendimiento y estabilidad.

Además, al ser un software de código abierto, ofrece acceso total al código fuente, a diferencia de NGINX, que no es completamente de código abierto, ya que NGINX, Inc. Introdujo la versión GINX Plus con características avanzadas como parte de su modelo de negocio, sus funcionalidades en monitoreo o balanceo de carga dinámico no estaría al alcance requerido.

HAProxy proporciona mayor flexibilidad y control para personalizar y adaptar el software según las necesidades que se requieran. Por último, es compatible con protocolos como HTTP/HTTPS (Maneja tráfico web, garantizando la seguridad de las comunicaciones), TCP (Proporciona equilibrio de carga en el nivel de transporte, permitiendo distribuir conexiones basadas en IP), SMTP/POP3/IMAP (Protocolo de correo electrónico), MySQL/Redis (Puede distribuir conexiones a estos servidores en ambientes de bases de datos). Lo que lo hace adecuado para los objetivos de este trabajo.

[26] Motor X, forma abreviada y estilizada de la palabra inglesa "Engine X". NGNIX Fue diseñado originalmente para resolver problemas de escalabilidad y estabilidad en servidores, Nginx se ha convertido en una opción popular para servidores web, servidores de aplicaciones, y balanceo de carga. [15]

a) Configuración completa

Se procedió a insertar el comando de la instalación directa de HAProxy en el sistema operativo.

$ sudo apt install haproxy

Una vez terminando la instalación de HAProxy, se procede a configurar el balanceador, se tiene que ir a la configuración y editar el algoritmo, agregar comandos extras al script original.

Para poder acceder a la configuración de HAProxy se necesita el siguiente comando.

$ sudo nano /etc/haproxy/haproxy.cfg

b) Descripción de la configuración del balanceador de carga

A continuación, se muestra el código por defecto que viene en el script.

```
global
        log /dev/log    local0
        log /dev/log    local1 notice
        chroot /var/lib/haproxy
        stats socket /run/haproxy/admin.sock mode 660 level admin
        stats timeout 30s
        user haproxy
        group haproxy
        daemon
defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http
```

A esta configuración por defecto se le tienen que agregar ciertas características para que funcione como un balanceador. Primeramente, se agrega lo siguiente.

```
frontend http_front
    bind *:80
    default_backend http_back
```

La página de HAProxy señala que “Una sección de **frontend** define las direcciones IP y los puertos a los que se pueden conectar los clientes. Puede agregar tantas secciones de **frontend** que se necesite para exponer varios sitios web o aplicaciones a Internet.” [16] En el puerto 80 se configura todas las conexiones por defecto del puerto 80 de HTTP y la otra parte del código quiere decir que todas las solicitudes las enviará al backend llamado http_back.

El siguiente segmento de comandos que se agrega, define la distribución de solicitudes.

```
backend http_back
    balance roundrobin
    server server1 192.168.1.64:80 check
    server server2 192.168.1.69:80 check
```

“Una sección de **backend** define un grupo de servidores a los cuales el balanceador de carga enrutará las solicitudes. Puede agregar tantas secciones de backend como necesite. Cada palabra de **backend** va seguida de una etiqueta, como **web_servers**, para diferenciarla de las demás. La etiqueta se utiliza principalmente para facilitar la lectura, pero también es útil cuando se hace referencia a tablas de barras y se categorizan métricas de tráfico. Debe constar únicamente de letras mayúsculas o minúsculas, dígitos, guiones, guiones bajos, puntos y dos puntos.” [17]

```
listen stats
    bind *:8080
    stats enable
    stats uri /stats
    stats auth [redacted]
```

Esta última sección de comandos es importante realizarla en caso de querer probar el rendimiento de un balanceador. “La página de estadísticas de HAProxy proporciona una fuente de datos casi en tiempo real sobre el estado de sus servicios proxy.” [18] Mediante el puerto 80

por defecto para el servicio HTTP donde se habilita y se configura con contraseña para poder tener acceso a la página, al escribir este código se activa la página de estadísticas.

c) Algoritmos de balanceo de carga

Un algoritmo de balanceo de carga es el conjunto de reglas que sigue un equilibrador de carga para determinar el mejor servidor para cada una de las diferentes solicitudes de los clientes. Los algoritmos de balanceo de carga se dividen en dos categorías principales. [19]

i) Método de carga estática

Los algoritmos de balanceo de carga estática siguen reglas fijas y son independientes del estado actual del servidor. Los siguientes son ejemplos de equilibrio de carga estática. [19]

ii) Método de turno rotativo

En el método de operación por turnos, un servidor de nombres autoritativo hace el equilibrio de carga en lugar de hardware o software especializado. El servidor de nombres devuelve las direcciones IP de los diferentes servidores de la granja de servidores por turnos. [19]

iii) Método de turno rotativo ponderado

En el equilibrio de carga por turnos ponderados, puede asignar diferentes pesos a cada servidor en función de su prioridad o capacidad. Los servidores con ponderaciones más altas recibirán más tráfico de aplicaciones entrante del servidor de nombres. [19]

iv) Método hash de IP

En el método hash de IP, el equilibrador de cargas lleva a cabo un cálculo matemático, denominado “hash”, en la dirección IP del cliente. Convierte la dirección IP del cliente en un valor, que luego se asigna a servidores individuales. [19]

v) Método de carga dinámico

Los algoritmos de equilibrio de carga dinámico examinan el estado actual de los servidores antes de distribuir el tráfico. Los siguientes son algunos ejemplos de algoritmos de equilibrio de carga dinámico. [19]

vi) Método de conexión mínima

Una conexión es un canal de comunicación abierto entre un cliente y un servidor. Cuando el cliente envía la primera solicitud al servidor, se autentica y establece una conexión activa entre sí. En el método de conexión mínima, el balanceador de carga comprueba qué servidores tienen la menor cantidad de conexiones activas y envía tráfico a esos servidores. Este método supone que todas las conexiones requieren la misma potencia de procesamiento para todos los servidores. [19]

vii) Método de conexión mínima ponderada

Los algoritmos de conexión mínima ponderada suponen que algunos servidores pueden gestionar más conexiones activas que otros. Por lo tanto, puede asignar diferentes pesos o capacidades a cada servidor y el equilibrador de carga envía las nuevas solicitudes de clientes al servidor con la menor cantidad de conexiones por capacidad. [19]

viii) Método de tiempo de respuesta mínimo

El tiempo de respuesta es el tiempo total que tarda el servidor en procesar las solicitudes entrantes y enviar una respuesta. El método de menor tiempo de respuesta combina el tiempo de respuesta del servidor y las conexiones activas para determinar el mejor servidor. Los equilibradores de carga utilizan este algoritmo para garantizar un servicio más rápido para todos los usuarios. [19]

ix) Método basado en recursos

En el método basado en recursos, los equilibradores de carga distribuyen el tráfico analizando la carga actual del servidor. Un software especializado llamado “agente” se ejecuta en cada servidor y calcula el uso de los recursos del servidor, como su capacidad de computación y memoria. A continuación, el equilibrador de carga comprueba si hay suficientes recursos libres en el agente antes de distribuir el tráfico a ese servidor. [19]

REFERENCIAS CAPÍTULO 2

[1] CLOUDFLARE, «¿Qué es SSH? Protocolo Secure Shell (SSH),» 2024. [En línea]. Available: https://www.cloudflare.com/es-es/learning/access-management/what-is-ssh/#:~:text=%C2%BFQu%C3%A9%20es%20el%20protocolo%20Secure,encriptar%20las%20conexiones%20entre%20dispositivos. [Último acceso: 3 noviembre 2024].

[2] TP-LINK, «TL-SF1008D Switch de escritorio de 8 puertos 10/100,» 2024. [En línea]. Available: https://www.tp-link.com/latam/home-networking/soho-switch/tl-sf1008d/#overview. [Último acceso: 24 agosto 2024].

[3] P. T. USA, «Alimentación a través de Ethernet/802.32az la transición ecológica de la alimentación a través de Ethernet (PoE),» PLANET TECHNOLOGY USA, 10 noviembre 2020. [En línea]. Available: https://planetechusa.com/802-3az-the-greening-of-power-over-ethernet-poe/. [Último acceso: 9 septiembre 2024].

[4] Raspberrypi.cl, «Raspberry Pi 4,» 2024. [En línea]. Available: https://raspberrypi.cl/raspberry-pi-4/. [Último acceso: 25 agosto 2024].

[5] C. M. R. S. J. S. J. R. G. y. A. C. C.N. Pérez, «INTRODUCCIÓN A LAS SINGLE BOARD COMPUTERS,» 1 Julio 2018. [En línea]. Available: https://www.boletin.upiita.ipn.mx/index.php/ciencia/771-cyt-numero-67/1539-una-introduccion-a-las-single-board-computers. [Último acceso: 25 agosto 2024].

[6] Kasperky, «¿Qué es un Firewall? Definición y Explicación,» AO Kasperky Lab, 2024. [En línea]. Available: https://latam.kaspersky.com/resource-center/definitions/firewall. [Último acceso: 5 noviembre 2024].

[7] Debian, «¿Qué es Debian GNU/Linux?,» Debian, 2024. [En línea]. Available: https://www.debian.org/releases/stable/s390x/ch01s03.es.html. [Último acceso: 24 agosto 2024].

[8] R. Pi, «¿Qué es Raspberry Pi?,» 2024. [En línea]. Available: https://raspberrypi.cl/que-es-raspberry/#. [Último acceso: 22 agosto 2024].

[9] P. Mullís, «Personaliza tu sistema operativo Raspberry Pi para el uso diario,» OpenSource, 24 junio 2020. [En línea]. Available: https://opensource.com/article/20/6/custom-raspberry-pi#:~:text=The%20%22Lite%22%20version%20of%20Raspberry,your%20custom%20Raspberry%20Pi%20OS.. [Último acceso: 22 agosto 2024].

[10] J. Pomeyrol, «Linux 6.6 es la nueva versión LTS,» MUYLINUX, 20 noviembre 2023. [En línea]. Available: https://www.muylinux.com/2023/11/20/linux-6-6-lts/. [Último acceso: 2024 agosto 23].

[11] Kernel, «Los Archivos del Kernel de Linux,» Kernel, 6 agosto 2024. [En línea]. Available: https://kernel.org/category/releases.html. [Último acceso: 24 agosto 2024].

[12] R. Hat, «¿Qué es el Kernel Linux?,» Red Hat, 27 febrero 2019. [En línea]. Available: https://www.redhat.com/es/topics/linux/what-is-the-linux-kernel. [Último acceso: 24 agosto 2024].

[13] A. H. S. PROJET, «Directivas comunes de Apache MPM,» Apache, 2024. [En línea]. Available: https://httpd.apache.org/docs/2.4/mod/mpm_common.html. [Último acceso: 27 agosto 2024].

[14] A. S. PROJET, «Módulos de Procesamiento MPMs,» 2024. [En línea]. Available: https://httpd.apache.org/docs/2.4/es/mpm.html. [Último acceso: 10 noviembre 2024].

[15] C. R. S., «Servidor Web Apache 2.4 (LINUX),» S.F. [En línea]. Available: https://wiki.cifprodolfoucha.es/images/2/2c/Servidor_Web_Apache_Sonia_Casas_Ramos.pdf. [Último acceso: 5 septiembre 2024]

[16] HAPROXY, «Ejemplo de configuración de Frontend,» HAPROXY, 2024. [En línea]. Available: https://www.haproxy.com/documentation/haproxy-configuration-tutorials/core-concepts/frontends/. [Último acceso: 27 agosto 2024].

[17] HAPROXY, «Backends,» HAPROXY, 2024. [En línea]. Available: https://www.haproxy.com/documentation/haproxy-configuration-tutorials/core-concepts/backends/. [Último acceso: 27 agosto 2024].

[18] HAPROXY, «Explorando la página de estadísticas de HAProxy (Lo que debe saber),» HAPROXY, 10 mayo 2019. [En línea]. Available: https://www.haproxy.com/blog/exploring-the-haproxy-stats-page. [Último acceso: 28 agosto 2024].

[19] AWS, «¿Qué es el equilibrio de carga?,» Amazon, 2023. [En línea]. Available: https://aws.amazon.com/es/what-is/load-balancing/. [Último acceso: 28 agosto 2024].

CAPÍTULO 3

HERRAMIENTAS USADAS PARA MEDIR EL RENDIMIENTO

3.1 Métricas

Las métricas son medidas cuantitativas que permiten monitorear y evaluar el rendimiento de los procesos de integración de datos y la calidad de estos [1]. Estas métricas son herramientas de análisis cuantitativo que facilitan la comparación y evaluación de características específicas de un conjunto de datos. Al permitir identificar la eficiencia, precisión y efectividad de las tareas de procesamiento, se convierten en herramientas ideales para monitorear no solo los procesos de datos, sino también sistemas operativos, servicios y aplicaciones. [2]

Existen diversos tipos de métricas, y cada una responde a una necesidad particular de análisis según lo que se quiera estudiar o entender. Algunos ejemplos incluyen métricas como:

- **Volumen de datos procesados:** Mide la cantidad de registros o filas que pasan a través de los procesos de integración. [1], [3]
- **Tiempo de procesamiento:** Cuantificar cuánto tarda un proceso específico en completarse. [1], [3]
- **Tasa de errores:** Indica la proporción de registros problemáticos en comparación con el total de registros procesados. [1], [3]

Las métricas pueden visualizarse en informes y paneles de control para monitorear continuamente el desempeño y la calidad de los datos, lo que permite a los administradores y analistas tomar decisiones informadas para optimizar los flujos de trabajo y mejorar la calidad de los datos. [3]

Para este proyecto, se utilizarán métricas obtenidas a través de herramientas como Netdata y, especialmente Siege. Estas métricas permitirán monitorear y evaluar diferentes aspectos de rendimiento y eficiencia, tales como el uso de CPU, memoria RAM y recursos de RED.

Con Siege, el enfoque es particularmente en evaluar la disponibilidad de recursos, el número de usuarios o instancias[27] activas que el sistema puede manejar simultáneamente, y el tiempo de respuesta bajo condiciones de carga. Esto proporcionará una visión más completa de la capacidad del sistema para gestionar demandas intensivas, ayudando a identificar posibles puntos de mejora en la administración de procesos y recursos generales.

3.2 Historia de Netdata

a) Historia del proyecto

Netdata fue fundada en 2013 por Costa Tsaousis, un ingeniero de software que identificó una necesidad crítica en el ámbito del monitoreo de sistemas: la falta de herramientas capaces de ofrecer una visibilidad instantánea y detallada del rendimiento en tiempo real sin consumir grandes recursos. Inicialmente se creó como una solución interna para su empresa. [4]

El proyecto ganó rápidamente interés en la comunidad de código abierto debido a su facilidad de uso, capacidad de autodetección de métricas y su enfoque en la eficiencia. Su base de datos personalizado optimiza el uso de CPU, memoria y almacenamiento, permitiendo un monitoreo continuo con un impacto mínimo en el sistema. A medida que evolucionó, Netdata integró funciones avanzadas como el análisis automatizado de anomalías, un motor de alertas configurable y un protocolo de transmisión de datos que soportan arquitecturas distribuidas. [4]

Características:

- **Monitoreo en tiempo real:** Permite a los usuarios ver el rendimiento del sistema de manera instantánea, detectando cuellos de botella y fallas rápidamente.
- **Bajo impacto en el rendimiento:** A pesar de su capacidad para recopilar y mostrar métricas en tiempo real, tiene un consumo de recursos muy bajo.

[27] Instancia: Proceso de ejecución o conexión independiente que simula el acceso de varios usuarios al servidor, realizando solicitudes de manera concurrente. [8]

- **Código abierto:** Es un proyecto de código abierto, con una comunidad activa de colaboradores y usuarios.
- **Interfaces avanzadas:** Ofrece una interfaz web interactiva y fácil de usar, que permite visualizar datos detallados con gráficos en tiempo real.

A lo largo de los años, continúa siendo desarrollado y mantenido por su comunidad y ha ganado popularidad. Hoy en día, Netdata es una solución que combina funcionalidades gratuitas de código abierto con servicios avanzados para usuarios empresariales debido a su facilidad de uso, capacidades avanzadas y su enfoque en el rendimiento y la eficiencia. [4]

b) Forma de uso

Netdata se basa en monitorear y analizar métricas de rendimiento del sistema en tiempo real mediante un panel de control accesible vía web. Una vez instalado en un servidor o dispositivo. [5], [6]

1. **Acceso al Panel de Control web.**

- Se accede a través de la dirección IP o dominio del servidor donde está ejecutando Netdata. Ver Apéndice A.
- El panel mostrará gráficos en tiempo real de métricas como: CPU, RAM, disco, red, procesos, entre otras métricas.

2. **Visualización de métricas.**

- **CPU:** Observa el uso en tiempo real por núcleo y el historial de carga del procesador.
- **RAM:** Evalúa la memoria disponible, en uso y caché del sistema.
- **Red:** Analiza el tráfico de entrada y salida, errores y paquetes perdidos.
- **Almacenamiento:** Revisa el uso de disco y la velocidad de lectura/escritura.
- **Procesos:** Monitorea los procesos activos y sus recursos asociados.

3. **Alertas automáticas.**

- Incluye notificaciones preconfiguradas para métricas críticas, por ejemplo, alta utilización de CPU o baja memoria disponible.

4. **Supervisión distribuida.**

- Compatible con arquitecturas donde se supervisan múltiples nodos desde un solo lugar. Esto es ideal para grandes infraestructuras.

5. **Resolución de problemas.**

- Identifica cuellos de botella en el rendimiento.
- Analiza picos de uso y tendencias a largo plazo
- Correlaciona métricas para diagnostica problemas complejos.

El enfoque principal de Netdata es proporcionar información clara, detallada y en tiempo real sin la necesidad de configuraciones complejas o interacción manual constante. Es una solución ideal para usuarios que buscan eficiencia y simplicidad en el monitoreo del sistema. [5], [6]

c) Métricas del CPU en Netdata

El CPU, la RAM y la RED, son elementos para comprender el comportamiento del sistema en entornos de carga variable, y su monitoreo permitirá identificar áreas de mejora y optimización.

i) CPU: Uso detallado del procesador de cada recurso utilizado.

El consumo de CPU se refiere a la cantidad de potencia de procesamiento que utilizan las aplicaciones que se ejecutan en su sistema. El system. CPU gráfico de Netdata representa la utilización total de CPU de su sistema Linux, desglosada en diferentes dimensiones. En este trabajo solo nos enfocaremos en **softirq**, **user**, **system**, **nice** e **iowait**, ya que cada dimensión proporciona información sobre cómo utilizan la CPU las distintas tareas y procesos. [5] [7]

1. **User**: Muestra el porcentaje de tiempo que la CPU pasa ejecutando procesos de usuario (es decir, aplicaciones que no son parte del núcleo). Indica cuánto tiempo está ocupada la CPU con tareas lanzadas por los usuarios.

2. **System**: Representa el porcentaje de tiempo que invierte en tareas del núcleo, como manejar llamadas del sistema, gestionar la memoria o controlar el hardware. Refleja el tiempo que la CPU está ocupada con procesos del sistema.

3. **Nice**: Indica el porcentaje de tiempo que la CPU usa para ejecutar procesos de usuario con una prioridad baja. Un valor alto podría señalar que las tareas de baja prioridad están utilizando una parte considerable del tiempo de CPU.

4. **Iowait**: Refleja el porcentaje de tiempo que está esperando a que se completen operaciones de entrada/salida, como el acceso a disco o red. Valores altos pueden sugerir que el sistema está limitado por E/S o que hay problemas con el almacenamiento.
5. **Irq**: Muestra el porcentaje de tiempo que la CPU emplea gestionando interrupciones de hardware, es decir, señales que los dispositivos envían a la CPU para ser atendidos. Valores altos indican que el sistema dedica mucho tiempo a eventos relacionados con el hardware.
6. **Softiqr**: Indica el porcentaje de tiempo que utiliza para procesar interrupciones de software, que son tareas del núcleo relacionadas con el hardware, como el manejo de paquetes de red. Valores altos podrían indicar que se invierte mucho tiempo en interrupciones de software.
7. **Steal**: Representa el tiempo que un hipervisor "toma prestada" (en entornos virtualizados) para otras máquinas virtuales en el mismo servidor físico. Valores altos de robo sugieren que hay competencia por los recursos entre las máquinas virtuales o que la asignación de recursos es insuficiente.
8. **Guest**: Muestra el porcentaje de tiempo que invierte en ejecutar procesos de máquinas virtuales en un entorno virtualizado. Valores altos de guest indican que una parte importante del tiempo de CPU se dedica a las máquinas virtuales.
9. **Guest Nice**: Representa el porcentaje de tiempo que se usa para ejecutar procesos de máquinas virtuales con baja prioridad.

3.3 SIEGE

a) Historia del proyecto

En el año 2000, Jeff Fulmer diseñó Siege como una alternativa más sencilla y accesible a las herramientas de pruebas de carga existentes, muchas de las cuales eran complejas y costosas. Concebida como un proyecto de código abierto, SIEGE se creó para permitir a desarrolladores y administradores de sistemas ejecutar simulaciones masivas de usuarios con el fin de evaluar la capacidad y el rendimiento de servidores, sitios web y aplicaciones. [8]

El objetivo principal es facilitar la simulación de tráfico intenso, recreando escenarios realistas en los que múltiples usuarios acceden simultáneamente a un recurso. Su diseño ligero y simple

permite generar grandes volúmenes de tráfico mientras se recopilan métricas como tiempos de respuesta, tasas de éxito y fallos, ayudando a identificar cuellos de botella o problemas de estabilidad en los sistemas bajo prueba. Soporta pruebas concurrentes y pruebas repetitivas para diferentes tipos de servidores, como HTTP, HTTPS y FTP. Además, es una herramienta flexible que puede integrarse en scripts, haciendo que sea ideal tanto para pruebas de rutina como para evaluaciones específicas de rendimiento. [8]

Características principales

- **Simulación de usuarios simultáneos:** Permite la simulación de miles de usuarios accediendo al servidor de manera concurrente[28].
- **Compatibilidad con varios protocolos:** HTTP y HTTPS, puede manejar protocolos como FTP.
- **Pruebas cíclicas:** Se puede configurar para ejecutar pruebas repetitivas, lo que ayuda a medir el rendimiento a largo plazo bajo distintas cargas de tráfico.
- **Simulación de tráfico real:** Siege genera tráfico que replica el comportamiento de usuarios concurrentes, ofreciendo datos realistas sobre el rendimiento del servidor.
- **Escalabilidad:** Permite evaluar cómo responde el servidor al aumentar la carga de usuarios concurrentes, identificando la capacidad máxima del servidor y preparar una infraestructura para el manejo del tráfico real.

Siege es una herramienta poderosa, sencilla y efectiva para realizar pruebas de carga, su capacidad para medir y reportar métricas críticas del servidor ayuda a manejar la carga sin comprometer el rendimiento. [8]

[28] Concurrente: Significa que todos los usuarios simulados están accediendo al servidor al mismo tiempo durante el periodo de prueba, enviando solicitudes de manera simultánea, no de forma continua o secuencia una tras otra. [12]

b) Forma de uso

Siege es una herramienta de prueba de carga que permite a los usuarios simular múltiples conexiones concurrentes para evaluar el rendimiento de servidores web, aplicaciones y sitios web. [8], [9]

1. La configuración básica

Siege permite personalizar las pruebas mediante parámetros como:

- **Número de usuarios concurrentes**: Representa la cantidad de usuarios simultáneos simulados en la prueba.
- **Duración de la prueba**: Define cuánto tiempo se ejecutará la prueba para recopilar datos de rendimiento.
- **Listas de URL**: Se pueden cargar desde un archivo externo para simular accesos a múltiples rutas en el servidor.

2. Ejecución de la prueba

Una vez configurados los parámetros, Siege genera múltiples solicitudes concurrentes al servidor. Durante la ejecución:

- Simula el comportamiento de usuarios reales interactuando con el sistema.
- Registra en tiempo real métricas como transacciones exitosas, fallidas, disponibilidad y el tiempo de respuesta promedio.

3. Recolección y análisis de resultados

Al finalizar la prueba, Siege presenta un informe detallado con métricas como:

- **Transactions**: Número total de solicitudes realizadas.
- **Availavility:** Porcentaje de disponibilidad del servidor durante la prueba.
- **Successful transactions:** Solicitudes exitosas completadas.
- **Failed transactions**: Solicitudes que no fueron atendidas.

Estas métricas se pueden usar para identificar el rendimiento máximo del servidor, los puntos de saturación y los errores específicos que requieren atención. Las cuales posteriormente se verán con más detalle. [8], [9]

c) Métricas que se usarán en SIEGE

Se trabajó con las métricas **transactions, availavility, successful transactions, y failed transactions,** lo que ayudará a evaluar el rendimiento, la estabilidad del servidor y del balanceador.

1. Transactions (Transacciones)

Es una solicitud HTTP, desde que se envía hasta que se recibe la respuesta, incluye la solicitud realizada al servidor (GET, POST, etc.) y la respuesta obtenida (como un código 200, 404, 500, etc.). El número total de **transacciones** indica cuántas solicitudes HTTP se enviarán al servidor durante la prueba.

2. Availavility (Disponiblidad),

Mide el porcentaje de transacciones que fueron exitosas y recibieron una respuesta válida del servidor.

3. Successful transactions (Transacciones exitosas)

Este es el número de transacciones que fueron completadas con éxito, es decir, que recibieron respuestas válidas del servidor.

4. Failed transactions (Transacciones fallidas)

Es el número de transacciones que no pudieron completarse con éxito, esto puede ocurrir por varios motivos:

- El servidor desarrolló un error como 404 Not Found o 500 Internal Server Error.
- Hubo problemas de conectividad (tiempo de espera o desconexiones).

Al ser un software de código abierto, sigue siendo una herramienta popular, especialmente en entornos Linux/Unix. [7], [8]

4. Interpretación de los resultados

Los resultados obtenidos deben interpretarse en función de los objetivos definidos al inicio de la prueba. Por ejemplo:

- Si se busca garantizar la estabilidad del servidor, un porcentaje de disponibilidad cercano al 100% indica que el servidor es confiable bajo la carga definida.

- Si lo que se quiere es identificar cuellos de botella, un aumento en las transacciones fallidas o en el tiempo de respuesta a medida que aumenta la carga puede señalar límites específicos que requieren optimización.

Después de interpretar los resultados, se pueden realizar ajustes en la configuración del servidor o en la arquitectura de la aplicación, y repetir las pruebas para validar las mejoras que se han implementado. [8], [9]

3.4 Línea base

La línea base es el valor de referencia que se establece como punto de partida para evaluarlo y dar seguimiento a un objetivo, esto permite detectar anomalías o cambios significativos en el rendimiento que requieren atención como:

- **Sobrecargas:** Un aumento anormal en el uso de CPU o memoria puede ser una señal de que el sistema está sobrecargado o de que algún proceso está consumiendo más recursos de lo habitual.
- **Tráfico en la red:** Un incremento en el tráfico de red fuera de lo común podrían indicar un intento de ataque o intrusión.
- **Problemas de rendimiento:** Si el sistema se vuelve lento y los tiempos de respuesta son más largos de lo que marca la línea de base, podría ser necesario ajustar u optimizar el rendimiento. [10]

3.5 Benchmark

Benchmark es una herramienta para realizar pruebas de rendimiento que se usa para evaluar y comparar la capacidad de sistemas, aplicaciones o componentes de hardware, midiendo su desempeño bajo ciertas condiciones de carga. Son esenciales para comprender cómo se comporta un sistema frente a un uso intensivo, ayudando a identificar cuellos de botella y áreas de mejora. Un benchmark realiza las siguientes operaciones: [12]

- Realiza pruebas durante un tiempo determinado utilizando la herramienta ab[29] y Siege.
- En cada prueba, va incrementando el nivel conexiones concurrentes hacia el servidor web.

[29] Apache Bechmark (ab) es un programa de comandos y se utiliza para realizar pruebas de rendimiento en servidores web, permite enviar un número específico de solicitudes HTTP a un servidor y medir su capacidad de respuesta. [12]

- Cada nivel de concurrencia examina múltiples recursos (URL) en el servidor evaluado.
- Al cambiar el nivel de concurrencia, se reinician los servicios implicados.
- Para cada nivel de concurrencia, se proporciona el promedio de respuestas por segundo obtenidas en el acceso a los distintos recursos. [11]

Tipos de Benchmark

- **Micro Benchmark**: Evalúa partes pequeñas de un sistema, como funciones específicas en el código.
- **Benchmark Sintético**: Estas pruebas simulan condiciones de uso intensivo que el servidor podría experimentar en el mundo real, aunque de forma controlada y predefinida evaluando el rendimiento. Aquí es donde entra Siege.
- **Kernel Benchmark** : Mide el rendimiento del núcleo del sistema operativo de un programa real. [12]

En nuestro caso se selecciona el uso de un Benchmark Sintético ya que nos interesa conocer el comportamiento del servidor web bajo una demanda de peticiones cercanas a la real y su posterior comportamiento al compartir la carga distribuida por el balanceador.

3.6 Establecimiento de la línea base de Netdata y Siege

La configuración utilizada para establecer la línea base de rendimiento tanto en Netdata como en Siege responde a la necesidad de realizar un análisis detallado y representativo bajo condiciones controladas que permitan evaluar la capacidad del sistema y el comportamiento del servidor al manejar distintas cargas.

1. **Configuración del servidor Apache**

   El servidor Apache fue configurado con un límite predefinido de usuarios concurrentes que puede atender eficientemente a 256 usuarios. Este límite establece la capacidad máxima del servidor en condiciones normales. La realización de pruebas busca sobrecargar el sistema deliberadamente para medir cómo responde el balanceador y servidor cuando excede su capacidad. Este enfoque proporciona una línea base que no solo mide el rendimiento, sino también los puntos en los que comienza a colapsar.

2. Pruebas para realizar a un solo servidor.

Se busca establecer el comportamiento de un solo servidor web bajo diferentes cargas o cantidad de peticiones realizadas al mismo. Se solicita la página web por defecto del servidor en cada usuario, lo que implica que el servidor estará ocupado cuando se requiera de nuevo el mismo documento.

De esta manera se puede establecer un perfil del rendimiento del servidor de acuerdo con el crecimiento de las peticiones y su posterior comparación con la respuesta dada por el balanceador.

3. Descripción del funcionamiento de instancias con usuarios

Para comprender el uso de esta herramienta, es entender cómo funcionan las instancias y su relación con los usuarios.

Una **instancia** se define como un grupo específico de usuarios que acceden al servidor de manera simultánea. Por ejemplo, si se configura una instancia con 100 usuarios, significa que en ese momento habrá 100 usuarios conectados al servidor al mismo tiempo.

El código desarrollado permite configurar tanto el número de instancias como el número de usuarios por instancia ver el apéndice C. Consideremos el siguiente ejemplo:

- Número de instancias: **10**
- Número de usuarios por instancia: **100**

$$10 * 100 = 1000$$

Esto significa que **1,000 usuarios estarían accediendo al servidor simultáneamente**.

Es importante comprender como funciona, ya que nos permite entender el funcionamiento en el rendimiento del servidor. A medida que incrementamos el número de usuarios totales, el rendimiento del servidor disminuye, pudiendo alcanzar un punto donde se generan errores. Este comportamiento es importante dimensionar correctamente para que el servidor pueda soportar la carga esperada. A continuación, se muestra una tabla comparativa de número de instancias contra usuarios.

Inst/Usua	10	30	50	100	200	250
10	100	300	500	1000	2000	2500
30	300	900	1500	3000	6000	7500
50	500	1500	2500	5000	10000	12500
75	750	2250	3750	7500	15000	18750

Tabla 3.1 Resultado de usuarios contra instancias.

Donde la casilla verde son los usuarios y en la casilla rosa son las instancias se multiplica cada casilla de usuario por instancia y nos da el resultado que muestra la tabla 3.1.

4. Medición de disponibilidad del servidor.

La prueba de carga realizada generó cierto nivel de estrés en un único servidor. El objetivo de estas pruebas es evaluar el rendimiento del sistema y determinar el impacto del tráfico generado en el servidor. Para ello, se elaboró y ejecutó un script[30] en bash[31] que permite especificar el número de instancias y la cantidad de usuarios por cada instancia La Figura 3.1 ilustra cómo este código gestiona la solicitud de instancias y usuarios, mostrando el funcionamiento del proceso.

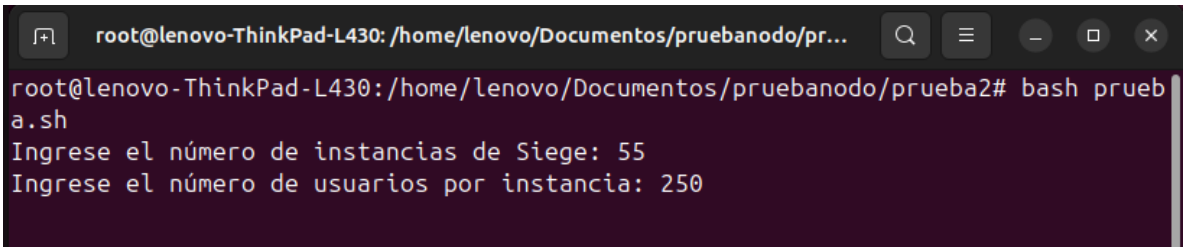

Figura 3.1 Ingreso de instancias y usuarios. [Figura generada en la terminal de Ubuntu].

[30] Script: Es un archivo de texto que contiene una secuencia de comandos que se ejecutan automáticamente para realizar una tarea específica o automatizar procesos en el sistema. [14]
[31] Bash: Es un intérprete de comandos que permite ejecutar secuencia de comandos (scripts) que el usuario escribe traduciéndolas para que el sistema operativo las ejecute, actuando como puente entre el usuario y el sistema. [14]

Después de ingresar los números de instancias de siege y el número de usuarios por instancia, el script bash comienza a ejecutarse. La Figura 3.2 muestra el funcionamiento de este proceso. Al momento de ejecutar el archivo, se muestra el número de instancia que está trabajando con los usuarios seleccionados. En esta ejecución, las instancias se inician de manera simultánea, lo que genera un mayor tráfico y aumenta la carga sobre el servidor.

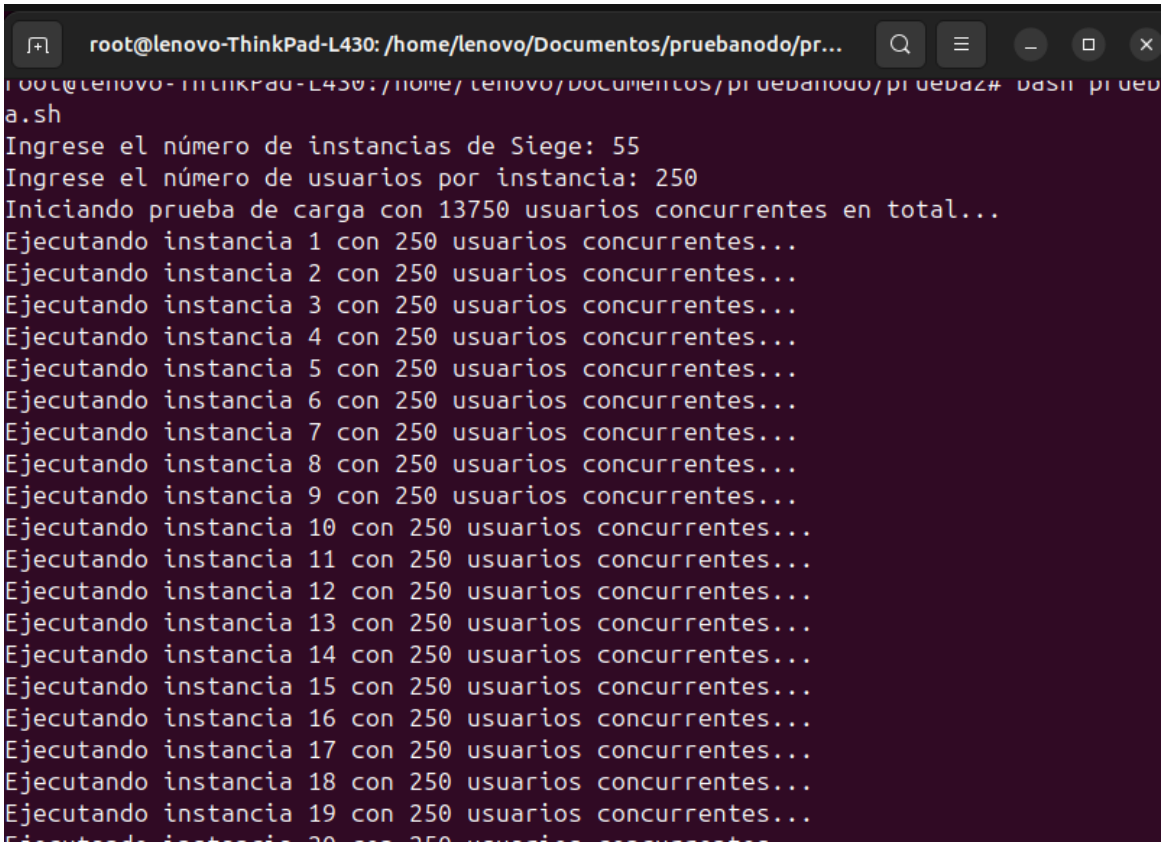

Figura 3.2 Muestra el funcionamiento del bash. [Figura generada en la terminal de Ubuntu].

Una vez terminado el bash se genera un archivo log[32] con información por cada instancia desarrollada, quiere decir que si son 25 instancias son 25 archivos, si son 125 instancias son 125 archivos.

[32] Archivo log: Es un archivo de registro que se genera después que termina de ejecutarse el bash, el cual registra información sobre la ejecución de un programa, como errores, advertencias y detalles de la actividad. [15]

Figura 3.3 Archivos generados con la información de cada instancia. [Figura capturada de los Archivos generados en documentos de Ubuntu].

Después de tener todos los archivos log de cada instancia queda explorar y empezar a recopilar datos tomando el promedio de porcentaje de cada prueba para poder graficar y tener un resultado de desempeño del servidor en la línea base establecida.

En la figura 3.4 **transactions**, índica el número total de transacciones que Siege ha realizado durante la prueba de carga, en este caso se enviaron y procesaron 3732 solicitudes HTTP al servidor web. **availability** mide el porcentaje de transacciones exitosas en comparación de las transacciones realizadas, por lo que 98.89% significa que casi todas las transacciones fueron exitosas y solo el 1.11% falló, la métrica indica que el servidor respondió correctamente bajo una carga específica, lo que es un buen indicador de rendimiento. **successful transactions** muestra que 3732 transacciones fueron exitosas, recibieron una respuesta válida del servidor, mientras que **failed transactions** índica el número de transacciones que fallaron durante la prueba y 42 transacciones no fueron completadas con éxito, pudo haberse suscitado por

problemas de conectividad, o sobrecarga del servidor. En general los resultados fueron positivos, ya que el servidor manejó la carga de solicitudes con éxito.

Figura 3.4 Información por instancia sobre el rendimiento del servidor. [Figura generada en la terminal de Ubuntu].

Estos parámetros permiten calcular el porcentaje de disponibilidad comparando las transacciones exitosas con las fallidas.

El 100% de disponibilidad se alcanza cuando todas las solicitudes procesadas son exitosas y no se registra ninguna solicitud fallida Un ejemplo de cuando no se alcanza esta disponibilidad sería si se registran 3732 exitosas y 42 fallidas. En este caso, el total de transacciones procesadas serían 3774. Para calcular el porcentaje de disponibilidad, se aplica una regla de tres, tomando las transacciones exitosas en relación con el total, lo que permite determinar qué porcentaje del servicio estuvo disponible sin errores.

$$\frac{3732}{42} = \frac{100\%}{X}$$

Ecuación 1

Primero se multiplica 42 por el 100 % nos da un resultado de 4200 después se tiene que dividir entre el número total de solicitudes, el cual es 3774, al aplicar la regla de tres queda el siguiente resultado. [16]

$$\frac{4200}{3774} = 1.11$$

Ecuación 2

El resultado, se resta al 100% de disponibilidad.

$$100 - 1.11 = 98.89$$

Ecuación 3

Como se puede ver, da como resultado 98.89 coincidiendo con los resultados lanzados por el programa SIEGE.

En esta instancia el resultado calculado coincide totalmente con la figura 3.4, quiere decir que la disponibilidad fue del 98.89%, teniendo un resultado óptimo en el servidor.

Cada instancia tiene ese porcentaje, si se ejecutan 5 instancias van a generar al mismo tiempo las solicitudes dependiendo de los usuarios lanzados. Por ejemplo, si se ponen las 5 instancias por 250 usuarios al multiplicarlo da el resultado de 1250 usuarios generando solicitudes al mismo tiempo y esto puede generar cierto estrés al servidor dependiendo del hardware del dispositivo.

Para evaluar el desempeño general, se recopilan los datos de cada instancia y se calcula el promedio, permitiéndonos observar cómo responde el sistema en una única prueba.

En general, se puede observar el estrés generado en el servidor y cómo responde ante el tráfico intenso, especialmente considerando que se utilizó un número elevado de instancias de Siege. Cabe reiterar que diversos factores influyen en estos resultados, incluyendo el hardware del servidor y el equipo utilizado para generar el tráfico.

Finalmente, se obtuvo el siguiente gráfico que muestra el comportamiento de un solo servidor bajo diferentes cargas y su disponibilidad relacionada.

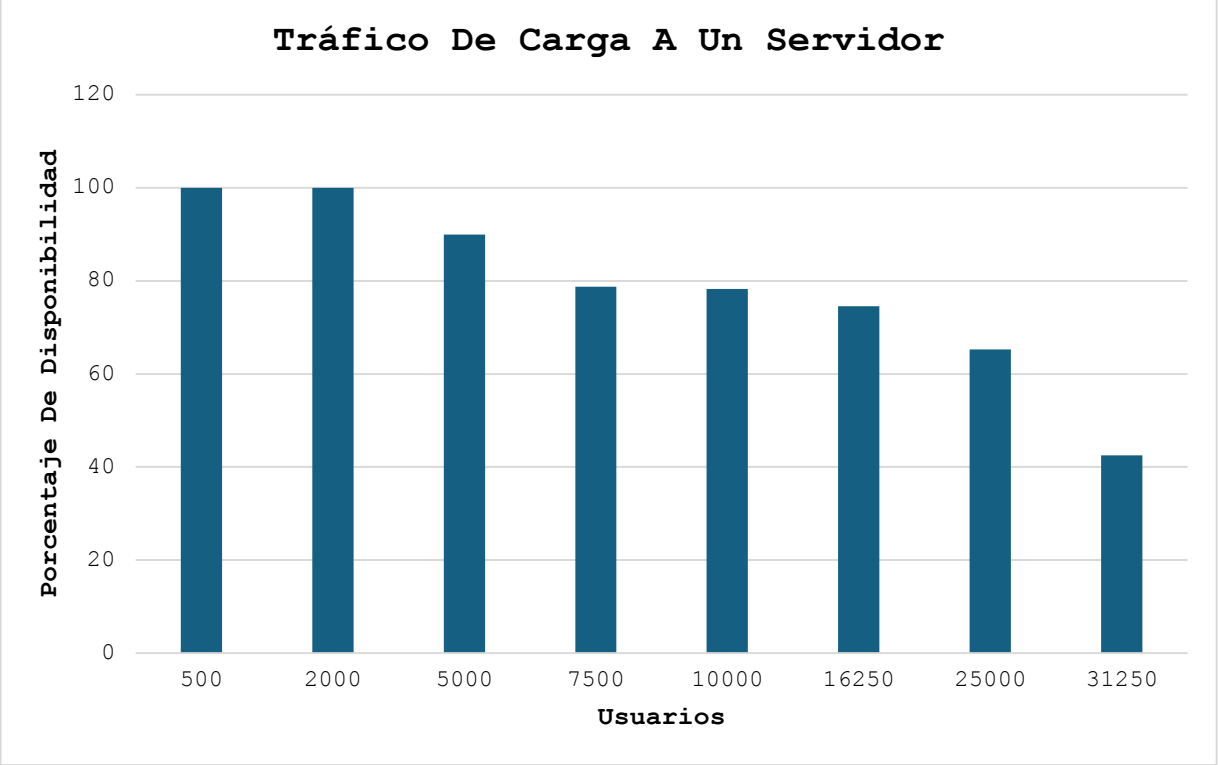

Figura 3.5 Línea base. Disponibilidad del servidor vs Cantidad de peticiones de usuario. [Elaboración propia].

3.7 Configuración de pruebas al balanceador

La misma cantidad de peticiones de usuarios que se dirigió a un único servidor se asignó al balanceador. Esto permitió evaluar el rendimiento del balanceador en comparación con el servidor único, se observó una diferencia significativa en el desempeño entre ambos.

Se realizaron las mismas pruebas graduales al balanceador, para después compararlos con la línea base

a) Medición de disponibilidad del balanceador

Al ejecutar las mismas pruebas se redirige la carga de tráfico al balanceador, y no a un solo servidor. Si no se cambia la dirección IP se podría volver a generar el mismo resultado que para un solo servidor. Para esto, solo es necesario actualizar la IP de destino en lugar de usar la IP del servidor **192.168.1.69** se debe usar la IP del balanceador, la cual se definió con **192.168.1.67** en la figura 3.6 muestra la línea donde se debe realizar el cambio para asegurar que el tráfico se dirija correctamente al balanceador.

Figura 3.6 Cambio de dirección de un servidor al balanceador. [Creación propia]

Redirigir correctamente el tráfico era importante para asegurar que el análisis fuera adecuado. Una vez realizada esta modificación, se aplicó la carga para efectuar la medición correspondiente.

REFERENCIAS CAPÍTULO 3

[1] I. Server, «Métricas,» IBM, 28 febrero 2021. [En línea]. Available: https://www.ibm.com/docs/es/iis/11.5?topic=methodology-metrics. [Último acceso: 3 noviembre 2024].

[2] Analitycs, «Dimensiones y Métricas,» Support Google, 2024. [En línea]. Available: https://support.google.com/analytics/answer/1033861?hl=es#zippy=%2Csecciones-de-este-art%C3%ADculo. [Último acceso: 28 octubre 2024].

[3] S. NET, «¿Qué es una métrica en informática?,» 2024. [En línea]. Available: https://servernet.com.ar/que-es-una-metrica-en-informatica/. [Último acceso: 15 noviembre 2024].

[4] NETDATA, «Detectives de seguridad,» Netdata, 1 abril 2024. [En línea]. Available: https://www.netdata.cloud/blog/safetydetectives/. [Último acceso: 2024 septiembre 2024].

[5] NETDATA, «Instalación del agente Netdata,» Learn Netdata Cloud, 2024. [En línea]. Available: http://surl.li/xvjoep . [Último acceso: 16 octubre 2024].

[6] NETDATA, «Guías de implementación,» Learn netdata cloud, 2024. [En línea]. Available: https://learn.netdata.cloud/docs/deployment-guides/. [Último acceso: 17 octubre 2024]

[7] NETDATA, «Consumo y utilización de la CPU,» 2 mayo 2023. [En línea]. Available: https://www.netdata.cloud/blog/understanding-linux-cpu-consumption-load-and-pressure-for-performance-optimisation/. [Último acceso: 19 septiembre 2024].

[8] GitHub, «SIEGE Instalar,» JoeDog, 2018. [En línea]. Available: https://github.com/JoeDog/siege/blob/master/INSTALL. [Último acceso: 25 octubre 2024].

[9] JoeDoge, «SIEGE,» 30 enero 2012. [En línea]. Available: https://www.joedog.org/siege-home/. [Último acceso: 25 octubre 2024].

[10] F. C. A. E. M. Cárdenas, Guía para el almacenamiento y cálculo de líneas base y metas, Ciudad de México: Coneval, 2019.

[11] S. d. r. central, «Estudio de rendimiento de servidores web,» 2017. [En línea]. Available: https://serviciosgs.readthedocs.io/es/latest/rendimiento/rendimiento.html. [Último acceso: 30 octubre 2024].

[12] U. E. d. M. Madrid, Benchmarck, Madrid: LAURETEA, 2018.

[13] Apache, «Directivas comunes de Apache MPM,» 2024. [En línea]. Available: https://httpd.apache.org/docs/trunk/es/mod/mpm_common.html#serverlimit. [Último acceso: 10 noviembre 2024].

[14] U. F. Profesional, «¿Qué es un bash script,» Unirfp, 1 septiembre 2023? [En línea]. Available: https://unirfp.unir.net/revista/ingenieria-y-tecnologia/bash-script/. [Último acceso: 5 noviembre 2024].

[15] ADOBE, «¿Qué son los archivos.LOG? ¿cómo abrirlos, verlos y editarlos?,» Adobe.com, 2024. [En línea]. Available: https://www.adobe.com/es/acrobat/resources/document-files/text-files/log-file.html. [Último acceso: 5 noviembre 2024].

[16] A. Alberti, «¿Cómo calcular la disponibilidad de una máquina?,» ALS Global, 24 agosto 2024. [En línea]. Available: https://www.alsglobal.com/es/news-and-publications/2022/06/como-calcular-a-disponibilidade-de-maquinas-e-equipamentos. [Último acceso: 14 noviembre 2024].

CAPITULO 4

COMPARACIÓN DE RESULTADOS

4.1 Análisis de carga de las métricas en el balanceador y servidor

Una vez que se generó el tráfico en la interfaz web de Netdata, se visualizó una serie de gráficos en tiempo real que muestran métricas detalladas sobre varios aspectos del sistema como se muestra en la figura 4.1

Figura 4.1 Monitoreo de métricas en tiempo real. **[**Figura generada con el software de monitoreo Netdata].

En la figura 4.2 la gráfica muestra el consumo de recurso del CPU de un servidor sin carga, no se ve ningún cambio.

Figura 4.2 Consumo de recurso del CPU del servidor sin carga. [Figura generada con el software de monitoreo Netdata].

En la figura 4.3 hay una carga de 55 instancias con 250 usuarios cada una a un servidor, en el cual se observa que **softirq** indica un 0% sugiere una carga sin interrupciones de software, si este porcentaje fuera demasiado alto indicaría una sobrecarga en el manejo de red o de otras tareas relacionadas con interrupciones y afectaría el rendimiento. **user** muestra 0.25% de tiempo que tarda el CPU en procesar las peticiones de los usuarios, un porcentaje alto indicaría una carga excesiva por peticiones del usuario. **system** indica el 0.75% como uso moderado de tiempo en tareas como del sistema, gestión de memoria y el control del hardware, si el valor fuera alto podría ser indicador de sobrecarga en funciones críticas del sistema y afectaría su estabilidad. **nice** muestra el 1 % esto sugiere que solo una pequeña fracción de tiempo se destina a tareas de baja prioridad, así que los procesos no críticos no interfieren con el rendimiento de tareas prioritarias, y por último **iowait** con el 0% indica que la CPU no experimenta demoras esperando por las operaciones de entrada y salida como el acceso al disco o la red, un valor alto podría indicar problemas de almacenamiento o de red, lo cual reduciría el rendimiento. Cada una de las métricas nos indican que el servidor está trabajando de una forma correcta, en el punto señalado como lo muestra la figura.

Figura 4.3 Consumo de recurso del CPU con carga durante un minuto, 55 instancias con 250 usuarios cada una. [Figura generada con el software de monitoreo Netdata].

El consumo total del CPU se mantiene bajo (sumando 3.0%), lo que evidencia que el servidor tiene suficiente capacidad para manejar la carga de trabajo generada por las 55 instancias y 250 usuarios sin experimentar saturación. Cómo lo muestra la figura 4.4

- La ausencia de consumo en **iowait** sugiere que el sistema está optimizado para evitar tiempos muertos en espera de recursos de entrada/salida.
- La métrica más alta, **System** 0.75%, representa un comportamiento esperado, ya que el servidor se centra en tareas de red y administración del tráfico, que recaen en el sistema operativo.

Estos resultados demuestran que el servidor está soportando la prueba y que aún dispone de capacidad adicional para manejar un incremento moderado en el tráfico sin comprometer el rendimiento.

Figura 4.4 Consumo de recurso del CPU del servidor en el balanceador con una carga de 55 instancias con 250 usuarios durante un minuto. **[**Figura generada con el software de monitoreo Netdata].

El consumo total del CPU se mantiene bajo (sumando 2.25%), lo que demuestra que el balanceador tiene la capacidad suficiente para gestionar la carga de trabajo sin experimentar saturación.

- La métrica más relevante es **System** 1.5%, como se esperaba, ya que las funciones principales de un balanceador están centradas en tareas de red y administración de tráfico, las cuales dependen del sistema operativo.
- La ausencia de consumo en **User** y **Softirq** refuerza que el balanceador está dedicado exclusivamente a sus funciones específicas, sin procesos de usuario o interrupciones de software que afectan su desempeño.

En general, los resultados indican que el balanceador está bien dimensionado y puede manejar cargas similares o mayores sin comprometer el rendimiento. Estos datos demuestran una configuración eficiente del balanceador para soportar el tráfico simulado sin consumir recursos de manera excesiva.

Figura 4.5 Consumo de recurso del CPU del balanceador con una carga de 55 instancias con 250 usuarios durante un minuto. **[**Figura generada con el software de monitoreo Netdata].

Al hacer la comparación visual con la gráfica del balanceador, se puede notar que tiene un mejor desempeño ante la carga, no obstante, eso se determinará más adelante con la comparación de resultados.

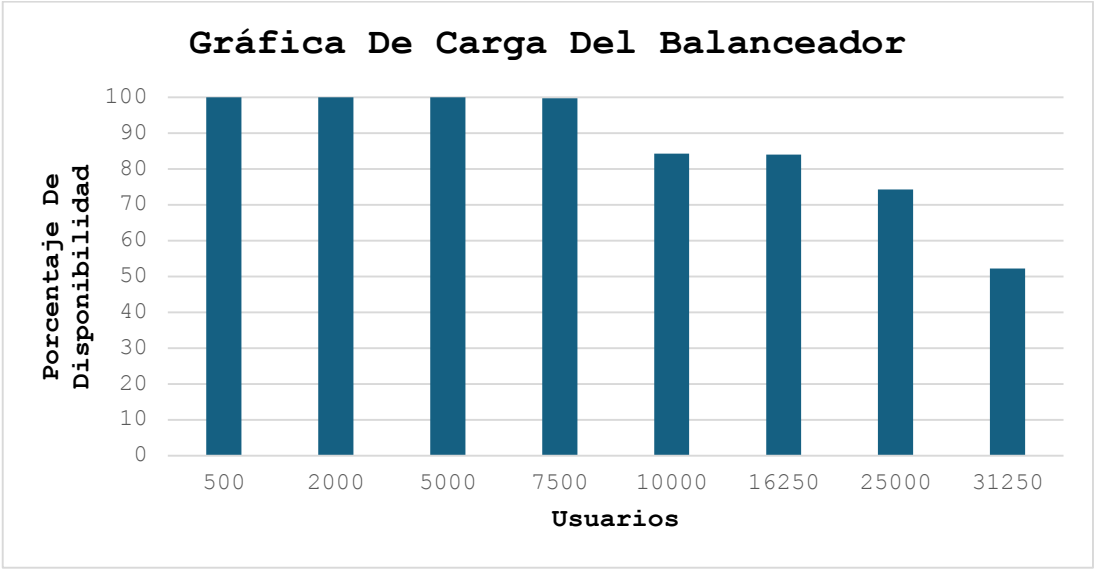

Figura 4.6 Desempeño del balanceador con carga. [Figura generada en Excel].

4.2 Comparación de resultados del balanceador y de un servidor

Tras realizar las pruebas y analizar los resultados, se procedió a comparar el rendimiento de un balanceador de carga frente a dos servidores. Esta comparación permitió identificar diferencias significativas en la distribución de la carga y en el manejo del tráfico, especialmente a medida que aumentaba el número de usuarios concurrentes.

Los resultados se plasmaron en gráficas detalladas que destacan estas diferencias de manera visual y cuantitativa. Por ejemplo, en la figura 4.7 se muestra cómo varían los porcentajes de disponibilidad al incrementar gradualmente el número de usuarios evaluados. Las barras azules representan el balanceador de carga, mientras que las barras naranjas ilustran el comportamiento de un único servidor.

A partir de los **5000 usuarios concurrentes**, se observa una diferencia notable en la disponibilidad del servicio. Mientras que el balanceador de carga mantiene un alto porcentaje de disponibilidad gracias a su capacidad para distribuir eficientemente el tráfico entre varios servidores, el rendimiento del único servidor comienza a disminuir drásticamente, presentando caídas en la capacidad de respuesta y un incremento en los tiempos de espera y errores.

Esta comparación no solo evidencia la ventaja del balanceador de carga en entornos de alta concurrencia, sino que también resalta su importancia como elemento para garantizar la escalabilidad y la estabilidad del sistema bajo condiciones de estrés.

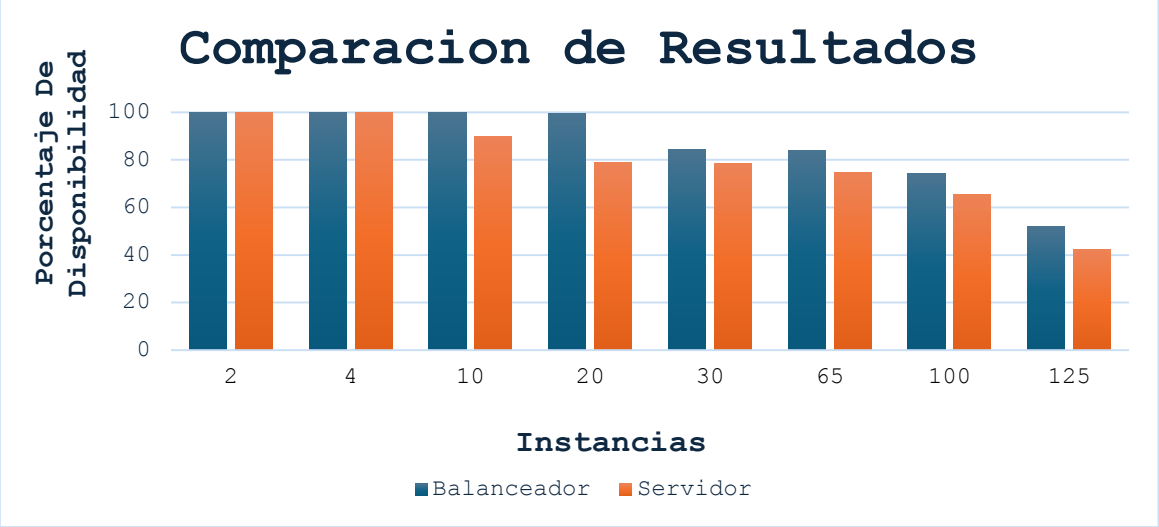

Figura 4.7 Comparación de resultados de balanceador y servidor.

[Figura Generada en Excel]

Las diferencias son evidentes visualmente, pero para un análisis más objetivo se puede consultar la tabla 4.1, donde los porcentajes destacan las variaciones de manera precisa

Usuarios	Balanceador	Servidor
500	100	100
2000	99.99	100
5000	99.95	89.95
7500	99.74	78.74
10000	84.25	78.25
16250	83.99	74.59
25000	74.29	65.29
31250	52.23	42.51

Tabla 4.1 Comparación de resultados.

[Tabla Generada en Excel]

Se puede notar que a los 16250 usuarios tiene menos disponibilidad, alrededor de un 11 % de diferencia, esto quiere decir que el balanceador es un apoyo para distribuir el tráfico en los dos nodos, a medida que aumenta el número de usuarios, el balanceador tiene más porcentaje de disponibilidad, quiere decir que la carga se distribuye entre los dos nodos y cumple con el objetivo que es balancear la carga del tráfico.

CAPÍTULO 5
CONCLUSIONES
Y
TRABAJO A FUTURO

5.1 Conclusiones

El presente trabajo permitió cumplir con el objetivo general de conocer un sistema balanceador de carga, su configuración y sus características en un entorno controlado, destacando las ventajas que ofrece en la expansión de la capacidad de atención de usuarios en un sitio web. La configuración y prueba del balanceador de carga se llevó a cabo mediante un ambiente local que incluyó herramientas como Netdata y Siege, permitiendo evaluar el sistema desde perspectivas tanto visuales como cuantitativas.

En relación con los objetivos específicos:

1. **Comprender el funcionamiento y las características del balanceador de carga.**

   Se logró identificar cómo el balanceador distribuye tráfico entre los nodos para prevenir saturaciones críticas y garantizar una mayor disponibilidad. El análisis de las métricas proporcionó información valiosa sobre la eficiencia del balanceador en comparación con un único servidor.

2. **Configurar un entorno de servidores distribuidos en una LAN.**

   Se implementó un entorno de servidores en red local que permitió simular escenarios reales de tráfico. Esta configuración sirvió como base para realizar pruebas con diferentes

volúmenes de usuarios, demostrando la importancia de la correcta sincronización entre nodos y balanceador.

3. **Evaluar el desempeño del balanceador en diferentes condiciones de tráfico.**

   Se realizaron a cabo pruebas con cargas progresivas, utilizando Netdata para el monitoreo en tiempo real de métricas como CPU (softirq, user, system, nice e iowait) y Siege para medir la disponibilidad, transacciones exitosas y fallidas. Estas pruebas demuestran que el balanceador distribuye la carga de manera eficiente, incluso bajo condiciones de alta demanda.

Aspectos destacados y limitaciones

El análisis evidencia que el hardware influye significativamente en los resultados. Las placas base de 8 GB y los procesadores limitados en las computadoras usadas para generar tráfico representaron un reto, especialmente durante pruebas de mayor intensidad. No obstante, estas limitaciones también resaltaron la capacidad del balanceador para adaptarse y mantener un rendimiento aceptable bajo restricciones de recursos.

Este trabajo no solo permitió alcanzar los objetivos planteados, sino que también ofreció una experiencia enriquecedora para comprender las capacidades y limitaciones de los balanceadores de carga. Herramientas como Netdata y Siege ayudaron abordar el problema contribuyendo a un análisis más general que sienta las bases para futuras mejoras en entornos de servidores distribuidos.

5.2 Mirando al futuro

El área de balanceo de carga y pruebas de rendimiento abre un amplio abanico de posibilidades para continuar explorando y optimizando el sistema. A continuación, se presentan diversas propuestas de trabajo a futuro.

1. **Implementación de Pruebas con Consultas a Bases de Datos**

   Realizar pruebas que no solo simulen tráfico hacia una página web, sino que también involucren consultas complejas y variadas a una base de datos. Esto implicaría:

- Diseñar consultas representativas de casos reales de uso.
- Medir el impacto de diferentes niveles de carga sobre la base de datos y el servidor de aplicaciones.
- Identificar cuellos de botella y optimizar tanto la base de datos como la página web para soportar estas cargas.

2. **Optimización del Uso de Recursos**

Investigar técnicas de optimización en el consumo de recursos, como:

- Configuración de límites de conexión en los servidores.
- Uso de compresión HTTP para reducir el tamaño de las respuestas.
- Configuración de políticas de caché para reducir la carga en el servidor.

3. **Implementación de Sistemas de Backups y Recuperación**
   Configurar copias de seguridad automáticas de la base de datos y los servidores, y realizar pruebas de recuperación ante desastres para garantizar la continuidad del sistema.

Perspectivas futuras

Este proyecto abre puertas a nuevas investigaciones y mejoras, como:

- Ampliar el análisis a más métricas, incluyendo RAM y red, para comprender aún mejor el desempeño del balanceador.
- Incorporar más nodos y usuarios en futuros experimentos para evaluar la escalabilidad del sistema en escenarios más complejos.
- Investigar configuraciones avanzadas que optimizan aún más la distribución del tráfico.

El presente trabajo ha permitido explorar aspectos relacionados con **“Instalación y configuración de un balanceador de carga para el manejo de varios servidores web”** sentando las bases para futuras investigaciones y desarrollos. Este cierre no marca un final, sino un punto de partida para nuevas ideas y colaboraciones que continúen enriqueciendo el campo.

APÉNDICES

APENDICE A

Forma de instalación de NETDATA

Paso 1

Ir a la página oficial de Netdata. https://learn.netdata.cloud/docs/netdata-agent/installation/linux/

Paso 2

Del lado izquierdo dar clic donde dice Linux, posteriormente copiar el script del recuadro en color negro y pegarlo en la terminal. Como se muestra en la figura A1.

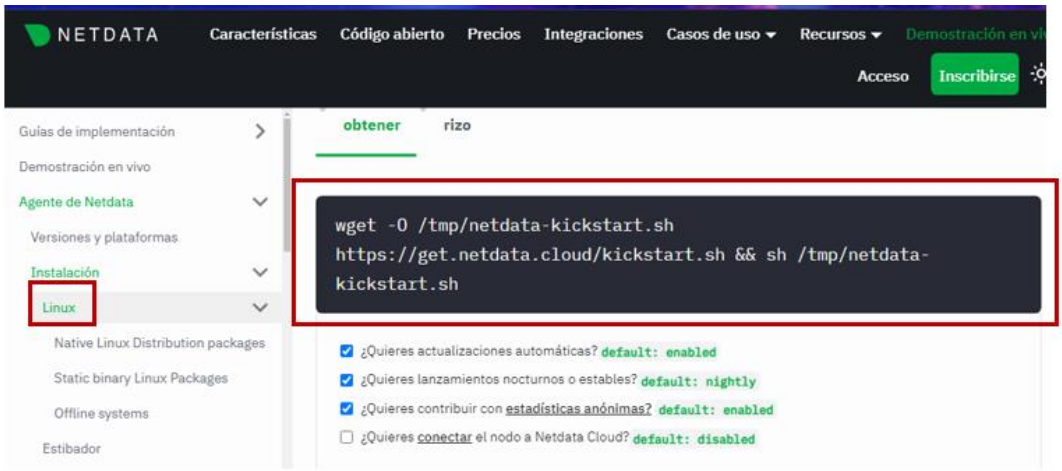

Figura A1 Instalación de Netdata. [1]

Este script hará lo siguiente:

- Instalará Netdata desde su repositorio oficial.
- Configurará los servicios para que se ejecuten automáticamente.
- Configurará el firewall, si es necesario.

Paso 3

Acceder a la interfaz web. Una vez instalado, Netdata debería estar corriendo y accesible a través de su servidor web local en el puerto 19999. Abra el navegador y vaya a:

http://localhost:19999

Si está instalado Netdata en un Servidor, remplace localhost con la dirección IP del servidor

http:// <IP_del_servidor>:19999

Paso 4

Para verificar que Netdata se esté ejecutando correctamente utilizando systemctl:

sudo systemctl status netdata

Si es correcto, deberá mostrar algo como esto:

```
● netdata.service – Real - time performance monitoring
  Loaded: loaded (/etc/systemd/system/netdata.service; enabled; vendor preset: enabled)
  Active: active (running) since ...
```

Figura A2 Verificación de instalación exitosa [1]

Netdata se actualiza automáticamente mediante el script de instalación. Si quieres actualizar manualmente en algún momento, solo necesitas volver a ejecutar el script de instalación.

APENDICE B

Forma de instalación de SIEGE

Paso 1

Ir a la página. https://blog.desdelinux.net/siege-medir-rendimiento-servidor-web/#google_vignette

Paso 2

Teclear en la terminal: **sudo apt install siege**

Forma de uso

Paso 1

Se utiliza un parámetro en la terminal para indicar el total de peticiones que hará y con otro se le indica la cantidad de peticiones simultáneas.

Por ejemplo siege c 50 r 100 http://sitio.com

Este script simula 50 usuarios concurrentes que acceden al sitio en 100 repeticiones

El output sería más o menos como lo muestra la figura B1.

```
Transactions:                4954 hits
Availability:               99.08 %
Elapsed time:               61.23 secs
Data transferred:           40.91 MB
Response time:               0.01 secs
Transaction rate:           80.91 trans/sec
Throughput:                  0.67 MB/sec
Concurrency:                 0.96
Successful transactions:     4954
Failed transactions:           46
Longest transaction:         0.11
Shortest transaction:        0.00
```

Figura B1 Output. [2]

Paso 2

Si se crea un archivo urls.txt y en él se ponen varias URLs del mismo sitio, se usa la siguiente línea para visitar esas URLs y medir el rendimiento con el siguiente script.

Siege .f urls.txt

-f indica que trabaje con una lista de URL dentro de un archivo de texto. El analizador urls.txt de siege admite comentarios variables.

Paso 3

El modo benchmark permite probar la capacidad máxima del servidor sin intervalos entre solicitudes. Para logar esto se escribe el siguiente script.

Siege -b http://sitio.com

-b inda a Siege que pase al modo de referencia, esto significa que no hay demora entre iteraciones.

La instalación de Siege depende del sistema operativo que se esté utilizando, en este caso se está usando Linux (Debian).

APENDICE C

Funcionamiento de algoritmo para generar tráfico virtual

En este apéndice se muestra el funcionamiento del código el cual se utiliza para generar el tráfico desarrollado, a continuación, se muestra por secciones del código para poder tener un mejor entendimiento de todo el código

En la figura C1 se muestra la primera parte del código la cual es importante ya que en esta parte del código recopila el número de instancias y la guarda en la variable **NUM_INSTANCES** y el número de usuarios **USER_PER_INSTANCE** que se va utilizar durante la ejecución de este mismo.

```
#!/bin/bash

# Leer el número de instancias desde el teclado
read -p "Ingrese el número de instancias de Siege: " NUM_INSTANCES

# Leer el número de usuarios por instancia desde el teclado
read -p "Ingrese el número de usuarios por instancia: " USERS_PER_INSTANCE
```

Figura C1 Primera parte del código del cual se encarga de recopilar datos. [3]

En la figura C2 se muestra la **URL** y la variable **DURATION** donde cada una tiene un valor en específico, la variable URL define el servidor el cual se le aplicará el estrés y la variable DURATION será el encargado de medir el tiempo que esté bajo estrés, por ejemplo, el tiempo se definió de 1M que esto equivale a 1 minuto.

```
# Variables de configuración
URL="192.168.1.64"
DURATION="1M"
```

Figura C2 Definición de la variable URL y DURATION. [3]

En la figura C3 se puede notar cómo es que hace una multiplicación de variables ya que multiplica las dos variables que se mencionaron en la Figura C1, las cuales son **NUM_INSTANCES * USER_PER_INSTANCE,** por ejemplo, si se tienen 10 usuarios y 5

instancias el resultado será de 50 usuarios por instancia, al hacer la multiplicación solo aparece la palabra completa, al ejecutar el programa, donde si tomamos el ejemplo anterior el renglón se escribe de la siguiente forma.

“Iniciando prueba de carga con 50 usuarios concurrentes en total…”

Posteriormente, se ejecuta el siguiente arreglo, que se incrementa de uno en uno hasta alcanzar el número de instancias definido por el usuario. Al entrar en este ciclo, cada número dentro del arreglo genera una leyenda correspondiente. Al final, el proceso redirige al comando SIEGE, lo que permite ejecutar las solicitudes en paralelo y comenzar la carga hacia el servidor. El comando encerrado en el círculo es el responsable de este efecto, ya que define la cantidad de usuarios por instancia, la duración de la prueba y luego retorna al arreglo. Esta sección se ilustra en la figura C3.

```
# Ejecuta instancias de Siege en paralelo
echo "Iniciando prueba de carga con $((NUM_INSTANCES * USERS_PER_INSTANCE)) usuarios concurrentes en total..."
for ((i=1; i<=NUM_INSTANCES; i++)); do
    echo "Ejecutando instancia $i con $USERS_PER_INSTANCE usuarios concurrentes..."
    # Redirigir la salida de cada instancia de Siege a su archivo de log correspondiente
    siege -c $USERS_PER_INSTANCE -t $DURATION $URL > "siege_instance_$i.log" 2>&1 &
done
```

Figura C3 Arreglo del código para desarrollar la carga paralela. [3]

El último párrafo es el wait, describe la espera a que termine todo el arreglo en la Figura C4 como se muestra a continuación.

```
# Esperar a que todas las instancias de Siege terminen
wait
echo "Prueba de carga finalizada."
```

Apéndice D

Configuración de SIEGE para Benchmark

En esta configuración es importante ya que funciona para tener un benchmark más eficiente, para esta tesis usamos la configuración que a continuación se explicara, primeramente, tenemos que utilizar la ruta adecuada y el archivo de configuración en este caso del software siege el cual iniciamos abriendo el terminal y tenemos que dirigirnos a la siguiente ruta.

Figura D1 ruta para acceder a la configuración de siege [Terminal], [4]

Una vez accediendo a la ruta que se eligió procedemos a entrar al archivo de configuración con el siguiente comando

sudo nano siegerc

una vez entrando en la configuración nos mostrara la siguiente descripción.

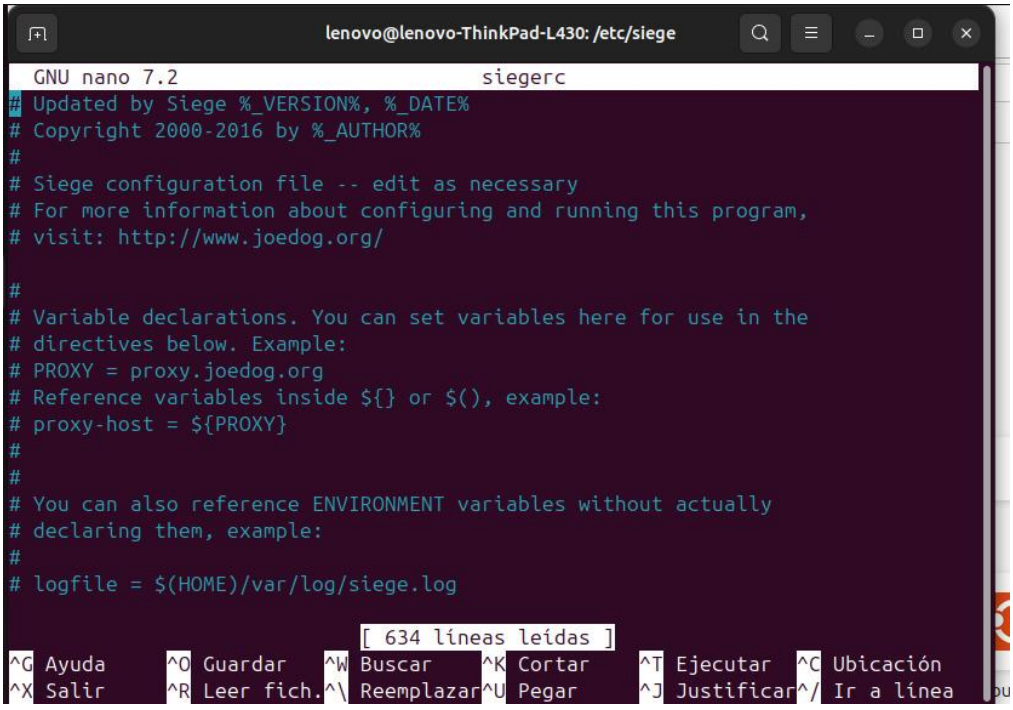

Figura D2 descripción de la configuración de SIEGE [Terminal], [4]

Este archivo llamado **siegerc** es de configuración de siege. La cual es una gran herramienta de pruebas de carga y estrés para servidores web. Se puede definir parámetros predeterminados para las pruebas evitando que se especifiquen manualmente cada que se quieran utilizar.

Las opciones que se utilizaron para el benchmark que se aplicó son

- **time = 30S**
- **concurrent = 50**
- **delay = 2**
- **benchmark = false**

donde cada una de ellas es una **FUNCION INDISPENSABLE** para que el benchmark obtuviera resultados óptimos a continuación se muestra la configuración de como quedo la configuración.

- **time = 1M**
- **concurrent = 50**
- **delay = 0**
- **benchmark = true**

en donde:

time: Define el tiempo de la duración de la prueba (segundos, minutos, horas).

concurrent: número de usuarios concurrentes, en esta sección se define cuantos usuarios se quieren definir que van hacer concurrentes.

delay: retraso de las solicitudes de cada usuario, esto quiere decir que cuánto tiempo se esperara en cada solicitud.

benchmark: esta función es justamente para no tener ningún retraso en ninguna solicitud y se lancen todas las solicitudes al servidor sin ningún tiempo de retraso al ponerle en true se activa esta función hasta cierto punto el delay no sería necesario, pero para asegurarnos lo definimos en cero, al terminar esta configuración está listo para iniciar el benchmark.

REFERENCIAS APÉNDICES

[1] NETDATA, «Instalar Netdata con kickstart.sh,» 2024. [En línea]. Available: https://learn.netdata.cloud/docs/netdata-agent/installation/linux/. [Último acceso: 25 Septiembre 2024].

[2] GitHub, «SIEGE Instalar,» JoeDog, 2018. [En línea]. Available: https://github.com/JoeDog/siege/blob/master/INSTALL. [Último acceso: 25 octubre 2024].

[3] GitHub, «How to get num_intances from output,» 2020. [En línea]. Available: https://github.com/facebookresearch/detectron2/issues/665. [Último acceso: 27 octubre 2024].

[4] WEBHOSTINGGEEKS, «How to Setup Siege to Perform a Stress Test on Linux (Ubuntu and CentoOS),» 2024. [En línea]. Available: https://webhostinggeeks-com.translate.goog/howto/how-to-setup-siege-to-perform-a-stress-test-on-linux-ubuntu-and-centos/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc. [Último acceso: 8 febrero 2025].