FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

Análisis de algoritmos balanceadores de carga para un clúster de servidores para mejorar la disponibilidad de un servidor

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

Autor(a) (es):
Bach. Fernandez Maluquis Jose Efrain
ORCID: https://orcid.org/0000-0003-1410-8912

Asesor(a):
Dr. Tuesta Monteza Victor Alexci
ORCID: https://orcid.org/0000-0002-5913-990X

Línea de Investigación

Infraestructura, tecnología y Medio Ambiente

Pimentel – Perú
2023

ANÁLISIS DE ALGORITMOS BALANCEADORES DE CARGA PARA UN CLÚSTER DE SERVIDORES PARA MEJORAR LA DISPONIBILIDAD DE UN SERVIDOR

Aprobación del jurado

MG. MEJIA CABRERA HEBER IVAN
Presidente del Jurado de Tesis

MG. MINGUILLO RUBIO CESAR AUGUSTO
Secretario del Jurado de Tesis

DR. TUESTA MONTEZA VICTOR ALEXCI
Vocal del Jurado de Tesis

DECLARACIÓN JURADA DE ORIGINALIDAD

Quien(es) suscribe(n) la DECLARACIÓN JURADA, soy(somos) **Fernandez Maluquis José Efrain** egresado (s) del Programa de Estudios de **La Escuela de Ingeniería de Sistemas** de la Universidad Señor de Sipán S.A.C, declaro (amos) bajo juramento que soy (somos) autor(es) del trabajo titulado:

ANÁLISIS DE ALGORITMOS BALANCEADORES DE CARGA PARA UN CLÚSTER DE SERVIDORES PARA MEJORAR LA DISPONIBILIDAD DE UN SERVIDOR

El texto de mi trabajo de investigación responde y respeta lo indicado en el Código de Ética del Comité Institucional de Ética en Investigación de la Universidad Señor de Sipán, conforme a los principios y lineamientos detallados en dicho documento, en relación con las citas y referencias bibliográficas, respetando el derecho de propiedad intelectual, por lo cual informo que la investigación cumple con ser inédito, original y autentico.

En virtud de lo antes mencionado, firman:

Fernandez Maluquis José Efain	DNI: 43758914	

Pimentel, 19 de agosto de 2023.

Dedicatorias

A Dios por ser mi fortaleza y guía en todo momento.

A mis padres, Mauro y Leonila por ser ellos quienes supieron guiarme día a día para ser lo que soy ahora, me impulsaron en todo momento el cual representa una gran alegría para ellos y para mí; por ese apoyo constante y sus ganas interminables, de animarme a seguir y a no desmayar en el intento.

A mi esposa Cinthia, por brindarme su amor y sobre todo su apoyo de manera incondicional, a mis hijas Camila y Maia por ser parte fundamental por ellas y para ellas este esfuerzo, ya que sé que siempre estarán conmigo en todo momento gracias.

El Autor

Agradecimientos

Quiero manifestar mi gratitud a Dios, quien siempre ha bendecido mi vida y de igual manera a toda mi familia por estar siempre presentes y que con su motivación me ha permitido realizar el presente proyecto de investigación.

De igual manera mis agradecimientos a la Universidad Señor de Sipán, a los profesores por compartir su experiencia y conocimientos me facilito que pueda crecer día a día como persona y profesional, logrando transmitirme sus conocimientos e inculcarme el deseo por la investigación, gracias a cada uno de ustedes por su tiempo, apoyo incondicional, dedicación y amistad.

Finalmente, al Mg. Víctor Tuesta Monteza por las orientaciones en el curso de investigación lo cual ha permitido llevar a cabo el presente proyecto de investigación.

El Autor

Índice

Índice de figuras

Índice de tablas

ANÁLISIS DE ALGORITMOS BALANCEADORES DE CARGA PARA UN CLÚSTER DE SERVIDORES PARA MEJORAR LA DISPONIBILIDAD DE UN SERVIDOR

Resumen

El aumento de la concurrencia a internet, ha generado un aumento de tráfico en la red a nivel mundial. La disponibilidad de los servicios depende de la capacidad de respuesta de los servidores, es así que las organizaciones han afrontado un aumento de peticiones de parte de sus usuarios que acceden a sus aplicaciones, generando una sobrecarga y reduciendo el tiempo de respuesta, ocasionando caídas de estos servicios. Ante lo mencionado, se realizará un análisis de algoritmos de balanceo de carga, más utilizados y de mejor desempeño en rendimiento y tiempo de respuesta, según la revisión sistemática de artículos, obtenidos de las bases de datos. Para lo cual se diseñará un clúster de servidor para implementar algoritmos de balanceo de carga, para mejorar la disponibilidad del servidor. Mediante las pruebas realizadas con las técnicas de balanceo seleccionadas, 3 escenarios (100, 300 y 500 usuarios) concurrentes y (500, 700 y 1000 solicitudes) respectivamente, ha permitido validar el desempeño en relación los indicadores definidos. En cuanto al tiempo de respuesta el algoritmo RR y HC tiene mejor desempeño en relación a WRR y LC, así mismo en cuanto al rendimiento el algoritmo WRR tiene mejores resultados en relación a RR, LC y HC, se concluye que si bien se puede afirmar que la aplicación de un clúster de servidores con algoritmos de balanceo de carga mejorara la disponibilidad de un servidor; también indicar que con el avance de las tecnologías se deben de ir evaluando nuevas tendencias tecnológicas.

Palabras Clave: Balanceo de carga, clúster, servidor, disponibilidad, rendimiento, concurrencia.

Abstract

The increase in Internet traffic has generated an increase in network traffic worldwide. The availability of the services depends on the response capacity of the servers, thus organizations have faced an increase in requests from their users who access their applications, generating an overload and reducing the response time, causing crashes. these services. Given the above, an analysis of the most used load balancing algorithms with the best performance in performance and response time will be carried out, according to the systematic review of articles obtained from the databases. For which a server cluster will be designed to implement load balancing algorithms, to improve server availability. Through the tests carried out with the selected balancing techniques, 3 concurrent scenarios (10, 30 and 50 users) and (100, 300 and 500 requests) respectively, it has made it possible to validate the performance in relation to the defined indicators. Regarding the response time, the RR and HC algorithm has better performance in relation to WRR and LC, likewise in terms of performance, the WRR algorithm has better results in relation to RR, LC and HC, it is concluded that although it can be stated that the application of a server cluster with load balancing algorithms would improve the availability of a server; also indicate that with the advancement of technologies, new technological trends must be evaluated..

Keywords: Load balancing, cluster, server, availability, performance, concurrency.

I. INTRODUCCIÓN

1.1. Realidad Problemática.

El acceso a internet, se ha convertido en un medio de intercambio de información en el día a día; los usuarios están conectados a internet mediante uno o más dispositivos, para revisar un correo, una página web, un curso, buscando información, juegos o contenido multimedia. A su vez las nuevas tecnologías han ido emergiendo como la computación en la nube, el big data, la integración de más dispositivos al internet mediante el IoT (internet de las cosas), los servicios SaaS, etc. Que de alguna manera contribuye a que más usuarios y empresas adquieran y hagan uso de estos servicios.

Estas nuevas tecnologías, ha generado que el tráfico de la red haya aumentado, al 2021, habían a 4.48 mil millones de usuario que acceden a Internet en todo el mundo, que se conectan desde diferentes dispositivos generan un aumento en las solicitudes de acceso a datos, imagen, video que están disponibles en la Web.[1]

En el Perú, a enero del 2021 había 19.90 millones de usuarios en internet, evidenciando un aumento de tráfico en los sitios web de empresas comerciales, de diarios importantes, entidades públicas y redes sociales.[2] También se ha presentado inoperatividad de los servicios digitales del banco de la nación publicado en El Comercio [3]. También se reportaron caídas de aplicativo del banco de Interbank el cual se mantuvo por varias horas [4].

Este comportamiento del aumento de usuarios que acceden a internet, llegan a generar que, Páginas web en de importantes medios de comunicación del mundo, de instituciones del gobierno como fue el caso de la Casa Blanca o el gobierno británico, y otras plataformas sufrieron importantes problemas de acceso [5]. Y sitios web para

consumidores de la banca presentaron dificultad para acceder o tardaban en cargarse, lo que generó preocupación a las personas que intentaban acceder a sus cuentas [6].
Según reportes de Dow Detector [7], los sistemas como Telegram, Twitter y Google, entre otros evidenciaron una sobrecarga ante la afluencia de los usuarios, ante caídas de Facebook y WhatsApp.

Asimismo, Barrantes [8], menciona que los servicios que proveen las empresas de internet en los últimos años han experimentado una mayor sobrecarga y generado la saturación de la red, asimismo los usuarios han percibido una demora en la respuesta.

Ante este comportamiento, el tráfico de la red aumenta y genera una sobrecarga de los datos, que superan el límite de capacidad de atención de los servidores, y experimenten los usuarios lentitud en las respuestas, si bien las grandes organizaciones pueden ampliar su arquitectura de servidor, para las medianas empresas por un tema de presupuesto quedan limitadas [9]

Por otra parte, Rohadi et al. [10] menciona que existen empresas e instituciones que tienen soluciones informáticas implementadas en un solo servidor y cuando se genera una sobrecarga de peticiones, este no podrá responder de manera adecuada, parte de este problema es que tiene un Único Servidor. A su vez otros factores como las especificaciones del servidor y una deficiente administración de la red.

Si bien para contrarrestar estos escenarios de volumen de carga, la solución era implementar un clúster de servidores estos deben complementarse con tecnologías o técnicas como los algoritmos balanceadores de carga, combinando el hardware y software en ambientes homogéneos y heterogéneos para medir el rendimiento [11].

Si bien existen una diversidad de algoritmos balanceadores de carga (como: least-connection, round robin, least-response-time, Hashing, entre otros.), aun así, es necesario realizar comparaciones integrando el uso herramientas como HaProxy, Heartbeat, ipvsmad, controladores OpenFlow, entre otros y analizar bajo diversos escenarios de prueba su desempeño a fin de implementar un clúster de servidor más eficiente, en su equilibrio y distribución de carga; de esta manera se logrará una mejor disponibilidad de un servidor.

Varios autores han realizado investigaciones, en las que realizan pruebas de balanceo es el caso de Saraswati et al. [9], en indonesia, en la que comparo el algoritmo Weight round-robin (RWW) con round-robin (RR), en 2 servidores web, con la herramienta de administración el ipvsadm para distribuir equilibradamente las solicitudes, donde la técnica de WRR tuvo mejor desempeño en tiempo de respuesta, rendimiento y error de atención.

También en Indonesia, Rohadi et al. [10]. utilizo la herramienta HaProxy y los algoritmos de Least Connection (LC) y Round Robin (RR), para medir cual técnica tenía mejor rendimiento y tiempo de respuesta, en el cual demostraron que el RR tiene mejor desempeño para ser implementado en un grupo de servidores web.

Del mismo modo, Liu et al. [11], en Taiwán, implementó una técnica de balanceo en una red definida por software (SDN) en ambiente virtual y real en arquitectura homogénea y heterogéneo, para distribuir la carga según la disponibilidad del servidor usaron el protocolo OpenFlow, se comprobó que la técnica de Least Conection (LC) tuvo mejor rendimiento que RR y WRL.

Esta nueva técnica SDN también fue aplicada por Jadhav, Mulla, and Narayan [12], de la India, con el fin de mejorar la gestión del tráfico pesado y si atender a más

clientes, realizando las pruebas en cuanto a cantidad de Host, Clientes y tiempo, en la cual la técnica de balanceo RR tuvo mejor desempeño que el algoritmo Random Load.

En cambio, Chiang et al.[13], de China, mediante el Algoritmo dynamic weighted random selection (DWRS) mejoró el equilibrio del clúster de servidor, para ello utilizó el controlador Floodlight quien recopila las cargas del servidor en tiempo real y así distribuir las peticiones al clúster de servidor más apropiado, evitando la sobrecarga.

En base a lo descrito, la presente investigación se propone realizar un análisis de algoritmos balanceadores de carga implementados en clúster de servidores para mejorar la disponibilidad de un servidor, teniendo en cuenta la realidad del país.

Saraswati et al. [9], en su investigación, Increasing web server performance using the web balancing Method, en Indonesia. Los servidores web cada día reciben un gran número de solicitudes, estos pueden atender según características que poseen, pero con el aumento del uso del internet el acceso a los sitios Web ha ido en aumento y no podrán soportarlo experimentando una sobrecarga, las grandes empresas puedan mejorar la arquitectura de servidor, pero las pequeñas empresas por su elevado costo están siendo afectadas, definiéndolo como el principal problema para aplicar una solución de balanceo de carga. Para lo cual aplicaron una tecnología de balanceo de carga mediante algoritmos como, round-robin (RR) y weight roundrobin (WRR), enviando cargas de peticiones al servidor sin hacer uso la tecnología balanceadores de carga y aplicando esta tecnología, para lo cual realizaron una investigación por fases para determinar el diseño y que componentes de hardware y software eran necesarios, llegando a utilizar un director o balanceador de carga, dos servidores web y mediante el balanceador web ipvsadm para distribuir uniformemente las solicitudes a cada servidor, midiendo los parámetros de tiempo de respuesta, error de solicitud y rendimiento. Como resultado de los 4 escenarios de 1000, 1500, 2000 y 5000 solicitudes que se realizaron, el mejor tiempo de respuesta

la obtuvo WRR con valores de 1.16 ms, 2.64ms, 4.92 ms y 28.75ms; en cuanto a rendimiento de E/S la obtuvo WRR con valores de 446.199 kbs, 469,38kb, 833,3 kb 923,95 kb /s., y en el Error de prueba también lo obtuvo WRR con 0.02%, 0.05%, 0.30% y 1.03%. Con los resultados se logró demostrar que el algoritmo de balanceo con el método WRR obtuvo mejor desempeño para ser aplicado en un ambiente de servidores cuyas características son diferentes.

Rohadi et al. [10], en su investigación, Clúster implementation on mini Raspberry Pi computers using Round Robin Algorithm, en Indonesia. Coinciden que en los últimos años el tráfico de información ha aumentado y que un solo servidor no satisface a las peticiones de los clientes y generando la disminución del rendimiento de un servidor lo cual hace los usuarios perciban tiempos de respuesta deficientes. Para lo cual se plantea comparar los algoritmos Round Robin con algoritmos de conexión mínima, para medir el parámetro de rendimiento del algoritmo RR, en relación a tiempo de respuesta y rendimiento, en un servidor web sin balanceador y en un clúster de servidor mediante el software de equilibrio HaProxy y los algoritmos de conexión mínima y por turnos, en cada escenario de prueba de algoritmo, se generaron clientes de 100, 200 y 500 cada uno enviando hasta 1000 solicitudes con el fin de que los resultados sean más exactos. En la prueba de servidor web único sin equilibrio de carga, la cantidad del cliente influye significativamente en el rendimiento, en cuanto a la pruebas aplicando balanceo de carga por turno y conexión mínima el método del Algoritmo de Roud robin permite mejorar la estabilidad del servidor mejorando la respuesta del servidor sin verse afectado por el número clientes, los valores de la prueba en cuanto tiempo de respuesta el RR obtiene 67.3ms, 95.3 ms y 182.3 ms, sobre 400ms,385ms y 550 ms del método de conexión mínima, en el rendimiento el método RR obtuvo valores de 62.50kbps, 76.42 kbps y 72.62 kbps, sobre los 32.50kbps, 46.42kbps y 55.40kbps del otro método. Según los resultados el algoritmo Round Robin, tiene disponibilidad para ser usado en grupo de servidores y sería una alternativa de solución de menor costo.

Noman and Jasim [14], Realizaron la investigación, A Comparative Performance Analysis for Static and Dynamic Load Balancing Techniques in Software Defined Network Environment, en Irak, Los balanceadores de carga tradicional, no son flexibles por su coste y dificultad para en la administración de una red, para tal fin buscan determinar la mejor técnica de equilibrio para mejorar en relación al rendimiento y escalabilidad de los servidores web. Para validar estas técnicas aplicaron un equilibro aleatorio, round-robin y round-robin ponderado mediante esquemas de equilibrio de carga estático y dinámico, centrándose en la comparación para medir de rendimiento de red y del servidor, mediante las técnicas de algoritmos de carga estáticos y dinámicos, para lo cual hicieron uso de la herramienta de HTTPerf, un controlador POX y comuntadores OpenFlow estos por ser de código abierto, en el cual se definen la lista de Número IP de cada servidor , creando una topología de red con el emulador de Mininet definiendo tres host que cumplen la función de servidores para los servicios web, y realizar las pruebas con él envió de hasta 100 conexiones /s, con una tasa de solicitud de 0 a 180 (req/seg), con un intervalo de 1/seg, cada 20 (req / eg. Logrando demostrar que los esquemas dinámicos tienen mejor rendimiento en comparación con los estáticos en un 7%, 2.2% y 2%, en cuanto al round-robin y round robin ponderado basado en el mínimo ancho de banda dinámico lograron mejor desempeño en un 8%, 3.3% y 2.56% en comparación con los esquemas de equilibrio de carga estática. Según los resultados se observa que el esquema de equilibrio de menor ancho de banda tuvo mejor resultado en la gestión del tráfico de red y del equilibrio de carga hacia los servidores.

Liu et al. [11], en su artículo, OpenFlow-based Server Cluster with Dynamic Load Balancing, en Taiwán, El big data, la computación en la nube y el rápido desarrollo de TIC han generado que en su momento los sistemas clúster fueron soluciones, afronten desafíos para atender las solicitudes y la seguridad en el centro de datos, y es de necesidad diseñar mejores estrategias de balanceo de carga para los ambientes de red

modernos. El estudio se centra en diseñar un nuevo mecanismo de carga dinámica basado en SDN no solo en ambiente virtual mediante Mininet, también en un entorno de hardware real con sistemas homogéneos y heterogéneos aplicando el protocolo OpenFlow, para que permita asignar las solicitudes según la disponibilidad del servidor y así reducir el cuello de botella que pueden generarse en el controlador Floodlight el cual fue modificado para aplicar la estrategia de carga mínima (LL). y compararlo con las técnicas de Round-Robin (RR) y Weighted Random Selection (WRL). Como resultado se evidenció que, en el entorno homogéneo, no hay mucha diferencia con los esquemas cuando la carga supera los 400 clientes, en un escenario de 300 clientes el LL es ligeramente superior que RR y RWS. En el entorno heterogéneo el LL mejoró su rendimiento en un 17.2% sobre el RR y un 21.4% sobre el WRS, esta diferencia se debe a que la estrategia LL elige el servidor más apropiado según la carga de procesamiento. Sin bien se demostró que la estrategia del LL basado en el marco SDN tuvo mejor rendimiento en un entorno heterogéneo, a futuro, planean aplicar su mecanismo en el entorno OpenStak.

Omer, Mustafa, and Abdalla [15], realizaron la investigación, Performance Analysis of Round Robin Load Balancing in SDN, en Sudan. En un entorno de servidores en el cual se tiene algunos con mejores características de CPU, RAM, etc. Los algoritmos balanceadores de carga no les asignan mayor carga generando que los servidores de menos recursos se puedan sobrecargar generando fallas mientras hay servidores inactivos que no se aprovechan sus especificaciones. Las pruebas se realizaron en entorno Ububtu, con el simulador Mininet usando SDN, mediante un controlador para controlar la red y medir los parámetros de transferencia, ancho de banda y fluctuación, parte aplicar el estudio se creó una red en el marco SDN, ejecutando código Python para medir (transferencia, rendimiento, fluctuación, retardo y pérdida de paquetes), luego la red se divide en dos grupos de 2 servidores c/u para las solicitudes TCP y UDP respectivamente. Según los resultados mediante el enfoque SDN, se evidencia que

mejoró el tiempo de respuesta del servidor, cuando se evalúa mediante TCP ejecutando un solo servidor, este es inestable con un rendimiento de 1,24 Mbit/s y aumenta a 12,1 Mbit/s cuando se cargan dos servidores. En la prueba para el tráfico UDP, el rendimiento al cargar un servidor es de 11,2 mbit/s y su rendimiento máximo es 86,9 Mbit/s, al duplicar el servidor el mínimo es 10,3 Mbit/s y el máximo es 89.5 Mbit/s. Finalmente se puede decir que el rendimiento de la red mejoró después de ejecutado el balanceador de carga SDN, teniendo un mejor rendimiento, transferencia, y disminución en el retardo, la fluctuación y pérdida de paquetes.

Rathore, Keswani, and Rathore [16], en su artículo Analysis of Load Balancing Algorithms Using Cloud Analyst, en la India, El consumo de servicios en la nube, a medida que los consumidores y solicitudes aumentan, se requiere del equilibrio de carga para garantizar el uso eficientemente de los recursos de este servicio. Por lo cual los autores se centran en un análisis de comparativo de los algoritmos Throttled, Round Robin, Ejecución de corriente distribuida equitativamente (ESCE), First Come First Serve (FCFS) y Shortest Job First (SJF) mediante pruebas simuladas en un escenario virtual utilizando Cloud Analyst, para la simulación se representa 6 usuarios y regiones y cuatro dentro datos, para medir los resultados, consideran los parámetros de tiempo de respuesta, promedio, tiempo de respuesta general, tiempo de servicio de solicitudes y tiempo de procesamiento en los centros de datos. Obteniendo que el tiempo de respuesta general el algoritmo FCFS tiene mejor rendimiento con un 70.89 ms a diferencia de los demás que superan los 100 ms, en tiempo de procesamiento el ESCE tiene menor tiempo en el proceso con un 0.41 ms superando a Throttled, a Round Robin que tienen 0.42ms y al Shortest Job Firs (SJF) con First Come First Serve (FCFS) que tienen los valores de 0.42 y 0.52 ms respectivamente. Finalmente, cuando se compara el tiempo de respuesta el FCFS funciona mejor, y en tiempo de procesamiento el ESCE muestra mejor desempeño, asimismo en su afán de mejor su trabajo se plantean ampliar su análisis con diversos parámetros y resultados de varias técnicas de equilibrio de carga.

Alankar et al.[17], en su investigación, “Experimental Setup for Investigating the E_cient Load Balancing Algorithms on Virtual Cloud”, en la India. Las soluciones de las nubes están siendo utilizadas cada día por usuarios por los servicios y aplicaciones web lo cual generan un elevado tráfico que hace necesario centrarse en la tarea de aplicar un equilibrador de carga bien automatizado. Realizaron un estudio para comparar los algoritmos más usados el Round Robin RR y Least Connections LC, en un entorno virtual con Cloud Analyst con un servidor de equilibrio de carga y varios servidores de servicios web en una topología paralela haciendo uso del servicio de HAproxy para los algoritmos y herramientas JMeter para poder realizar el estudio de rendimiento de estrés y carga, aplicando 4 pruebas con usuario de 1500, 3000, 4500, 6000 dividido en 3 grupos de muestras c/u que llegan al servidor en un lapso de 20s, 25s, 1m y al mismo tiempo respectivamente. Los valores obtenidos de las pruebas en el parámetro de tiempo promedio del algoritmo round robin es de 236, 362, 452 y 144 ms, en cambio los tiempos mínimo de conexión fue de 271,590,762 y 176 ms, en cuanto al tiempo de rendimiento el valor más alto la obtuvo RR con 58.53m, 106.37, 156.21 y 194.14 ms sobre los 57.03, 102.06, 132.13 y 182.92 del algoritmo de conexión mínima y en cuanto al ancho de banda de la red en el envío y recepción de kb/s el RR obtuvo un mejor desempeño con valores recibidos de 1371.83, 2469.61, 4006.99, 3954.47 y envió 7.37, 13.85,21.66 y 25.53 kb/s, a diferencia del Algoritmo de menos conexiones los kb/s recibidos fueron de 1335.37, 2369.54, 3389.18 y 3726 logrando enviar 7.02, 12.99, 17.94 y 23.52 kb/s. Si el tiempo de respuesta promedio general es mínimo y el rendimiento es alto, entonces el algoritmo de operación por turno RR tiene un mejor rendimiento. A futuro se desea comparar el rendimiento de algoritmos de carga en servidores de BD.

Aymaz et al. [18], en su investigación, An Analysis of Load Balancing Strategies withWireshark in Software Defined Networks, en Turquía. El aumento de uso del internet, arquitecturas de red convencionales son complejas y con dificultades para su gestión.

Mejorar estas limitaciones mediante la aplicación del marco SDN aplicando estrategias para decidir qué servidor atiende a las solicitudes, analizando con el analizador de protocolos Wirehark la estrategia de equilibrio de carga como Randon y round robin, aplicado en una topología creada en Mininet, el controlador POX mediante el lenguaje de programación Python. Se interconectan 4 host, un controlador y un conmutador OpenFlow, de los cuales el H1 y h2 son servidores de equilibro de carga que mediante los comandos de Python atenderán solicitudes de los otros host H3 y H4.00. Según las pruebas para el tiempo de respuesta mediante de equilibro de carga aleatorio el servidor H1 y H2 tiene una distribución desequilibrada, En la prueba de balanceo por turno, las solicitudes se distribuyó por igual, aun así tiene la desventaja que distribuye por no tener en cuenta la carga de las solicitudes; para medir el tráfico de red se evidencio el mismo comportamiento que en la prueba de equilibrio aleatorio el tráfico no se distribuyó por igual, en cambio en la prueba por turno el Servidor H1 y H2 respondieron la misma cantidad de paquetes. El nuevo enfoque en SDN ha permitido comprobar mediante sus estrategias para la distribución de la carga, como Random y round robin, que el uso de Wireshark facilita la supervisión de la red.

Hamdani, Aklouf, and Chaalal [19], realizaron la investigación, A Comparative Study on Load Balancing Algorithms in Cloud Environment, en Argelia. Ante el emergente modelo de la computación en la nube, se vienen aplicando varios métodos para mejorar el acceso a los servicios y no afecten a la plataforma. Este estudio se basó en realizar pruebas midiendo los parámetros de tiempo de respuesta, procesamiento y costo de la Máquina virtual y transferencia de datos, mediante tres algoritmos comunes como es el Round Robin, Throttled y Equal Spread Current Execution con la ayuda de CloudAnalyst que facilita un entorno de simulación realista, aplicado mediante el lenguaje Java, se configuraron seis bases de usuarios en seis regiones diferentes. Se obtuvo que el algoritmo THR utiliza un menor tiempo a comparación de los otros, tanto en el tiempo de respuesta general, tiempo de procesamiento del CD, los tiempos de servicio de solicitud

del CD y el costo de procesamiento. Por lo tanto, permite reducir el tiempo del servicio de solicitud de datos del CD y el uso de recursos, En cuanto a los costos se ha demostrado que las técnicas de equilibro de carga no afectan en el costo de procesamiento. Asimismo, precisan que los resultados pueden variar según la configuración que se realice a los parámetros, para futuro pretenden aplicar el estudio en un entorno de redes neuronales.

Chiang et al. [13] en su artículo, SDN-based server clusters with dynamic load balancing and performance improvement, en China, Proporcionar servicios con disponibilidad y escalabilidad para lo cual se debe tener un equilibrio de carga en tiempo real del Lado de servidor y mejorar el rendimiento de un clúster de servidor que atienden diversas capacidades de procesamiento. Aplicar un algoritmo dynamic weighted random selection (DWRS), que considera la carga en tiempo real, modificando el flujo y mejorándolos con la implementación de subprocesos múltiples para la seleccionas más apropiada de servidor y evitar la sobrecarga, mediante el controlador Floodlight, el conmutador OpenFlow y varios servidores que contienen varios host que forman un clúster, aplicando en ambientes homogéneos y heterogéneos con conexiones simultáneas de 200 a 1000, el DWRS recopila las cargas en tiempo real mediante el Floodlight para analizar la carga en el servidor que varía de 0 a 100 mediante el uso de demonios en cada servidor y asignar las solicitudes en base al peso del servidor evitando un desequilibrio de carga. En comparación con otros algoritmos como least loaded, the least connections, and the random selection, en ambientes homogéneos las de pruebas de 400 conexiones el DWRS tuvo mejor desempeño en un 7.1% que el RR, en 600 y 800 conexiones aún supera al RR en un 6.4% y un 4.8%, a su vez los resultados de otros algoritmos son similares en 1000 conexiones, en cuanto al ambiente heterogéneos mediante el mismo procedimiento de conexiones el DWRS tiene mejor rendimiento que el RR en un 7.5% y 12%. El algoritmo propuesto funciona en una arquitectura de controlador para un clúster de servidor a gran escala basado en OpenFlow. En próximos

estudios desean analizar los controladores distribuidos para mejorar su escalabilidad en una agrupación de clúster de servidor.

R. Li, Li, and Li [20], en la investigación, An Integrated Load-balancing Scheduling Algorithm for Nginx-Based Web Application Clusters, en China. En la tecnología Docker, el desarrollo de las aplicaciones tiende a depender de sus contenedores por su escalabilidad y la mayoría de aplicaciones Web se implementan en plataforma de la nube y se utiliza el enfoque en clúster con balanceadores de cargo como nginx, haproxy, etc, frente a clúster Web los balanceadores de carga solo establecen el equilibrio en base al rendimiento inicial y no en función al estado carga en tiempo real. Se propone el algoritmo de Dynamic Weighted Least-Connection (DWLC) y el Apache Beach para medir el rendimiento de servidor Web HTTP, de esta manera simular una cantidad de subprocesos simultáneos hacia servidor nginx y 5 de trabajo, para medir el ancho de banda entre ambos se simuló con 100 usuario para 10000 procesos simultáneos y así poder monitorear adecuadamente y hacer seguimiento del estado del host y los contenedores. Según las pruebas se estableció el umbral de peso en 5 con intervalo de 3s, para comparar la velocidad de respuesta de los algoritmos, en el cual el DWLC tiene mejor respuesta con 52.6% y 46.4% sobre el RR y LC, cuando se completa el 95% de solicitudes. Se evidencia que el del algoritmo DWLC contribuye de manera eficiente en el tiempo de respuesta del clúster web a diferencia de RR y LC.

Jadhav et al. [12], en su investigación, An Efficient Load Balancing Mechanism in Software Defined Networks, en la India. Al internet se accede desde cualquier lugar, han aumentado los usuarios, las redes van creciendo en base a nuevas tecnologías, lo que genera dificultades a la hora de gestionarlas en el tráfico pesado y atender a este gran número de clientes. Se considera el uso de controladores SDN que permiten aplicar mecanismos de equilibrio automatizados en ese contexto se aplica la estrategia de balanceadores de carga Round Robin y Random load con OpenFlow, usando el emulador

Mininet para medir los parámetros de rendimiento y pérdida promedio de paquetes, comparando estos algoritmos en 3 escenarios. En escenario variable número de host utilizaron 3, 6 y 9 servidores, para calcular el promedio de rendimiento y pérdida de paquetes, obteniendo mejor rendimiento el RR en comparación al Random Selection, en el que la carga en las rutas disminuye y la respuesta es más rápida, en el escenario variable de número de clientes de 6, 12, 24 y 48 respectivamente, el rendimiento disminuye afectando a la red, a pesar de ello el Algoritmo Round Robin Selection tiene mejor performance que Random Selection, finalmente, en la variable tiempo de 20,40, 60 y 80 s respectivamente, la pérdida de paquetes es menor, al de flujo creciente se observa que (RR) tiende a ser más eficiente que el Random load balancing. El modelo SDN tiene los requisitos para actuar dinámicamente en su aplicación para mejorar los recursos de la red.

Wilson Prakash and Deepalakshmi [21], realizaron la investigación, DServ-LB: Dynamic server load balancing algorithm, en la India. En estos últimos años, el acceso de los clientes a los servicios Web ha ido en aumento, lo que genera más tráfico en red, en el clúster se puede agregar o quitar servidores, pero su administración se hace complicada, y estos servidores web enfrentan varios problemas entre ellos la sobrecarga. Por lo cual la técnica de equilibro de carga ayudan a resolver los problemas de tráfico, por lo que se propuso un Algoritmo de equilibrio de servidor dinámico (Dserv-LB), que trabaja con los conmutadores OpenFlow basado en SDN, y mediante el Mininet se simuló una topología, con un Host en Ubuntu y controlador POX, en la cual se implementa los algoritmos random, round-robin y DServ-LB, a su vez se amplió la red la red con Maxinet utilizando el controlador Docker como anfitrión, 4 conmutadores OpenFLow, dos computadoras como POX y 4 computadoras como servidor web que ofrecen el mismo servicio, para medir la carga se usó el protocolo sFlow, aplicando una carga de 500 solicitudes por segundo. Los resultados del parámetro de carga de servidor, el algoritmo Dserv-LB tuvo mejor rendimiento a comparación con Random y RR, debido a que el

propuesto utiliza el valor de uso CPU como parámetro para medir la carga de servidor lo cual le permite distribuir de manera uniforme las solicitudes, en cuanto a uso de la memoria tiene el mismo comportamiento, en la tasa de caída de solicitudes el DServ-LB tiene menor valor que el de carga aleatorio y RR, en el tiempo de respuesta el propuesto tiene mejor resultado. Asimismo, se comparó el rendimiento al utilizar un controlador único y múltiple. Según el resultado se evidencia que el algoritmo propuesto tiene un buen rendimiento en servidores web, en base a la comparación con los algoritmos existentes. A su vez en el futuro se pretende aplicar en servidores de la nube.

Suherman, Aziz, and Nababan [22], realizaron la investigación, Load balancing algorithm for a local video network, en Indonesia, Con la implementación de nuevas de aplicaciones y el incremento de las solicitudes de los clientes aumentan, supera el límite del servidor, este se sobrecarga y el servicio que ofrece tienden a detenerse, el clúster es una solución, pero emplean un balanceador para distribuir las tareas y la técnica de manera secuencial hace que el servidor una vez que completa su tarea debe de esperar su turno para atender otra solicitud por lo que el clúster no estaría funcionando de manera óptima. Para evaluar el rendimiento, se realiza mediante la programación de socket en Java, en una red experimental con dos Host como servidor y tres hosts como clientes, cada cliente generará en un rango de 5 a 10 solicitudes, las pruebas se aplican con el algoritmo round robin (RR), Weighteg Roun Robin (WRR), source IP Hash and random, midiendo el parámetro de rendimiento se mide la fluctuación y la pérdida de paquetes, para el experimento se replicaron 100 veces para cada algoritmo. En los resultados se observa que el retardo de los paquetes varía entre cada prueba, estos retardos llegan a los 27ms y 45ms, teniendo en cuenta que en tiempo real los datos necesitan de 150ms para lograr el mejor rendimiento, el método aleatorio genera un retardo de 45.75ms, por lo que el WRR tiene el valor más bajo con un 38.85ms, seguido de IP, RR y aleatorio. En cuanto al promedio de retardo, pérdida y fluctuación el WRR tiene el mejor rendimiento

con 38.5%, 4.58% y 1.85ms respectivamente. Que el WRR tiene mejor rendimiento que los otros algoritmos, por lo que el WRR es más rápido en elegir y asignar el mejor servidor ante las solicitudes de los usuarios.

Arahunashi et al. [23], en su investigación, Implementation of Server Load Balancing Techniques Using Software-Defined Networking, en la India. El alto crecimiento de los usuarios conectado a internet, ya sea por las redes sociales, transmisiones de video, descargar de información lo cual conlleva a que el tráfico de red se congestione y como consecuencia se sobrecargue el servidor, en la cual a pesar de tener técnicas de red tradicionales de balanceo no son capaces de gestionar las sobrecargas actuales de la red. Para afrontar estas debilidades se pueden realizar mediante el uso de Software Defined Networks (SDN) que no integra el plano de control y el de datos, por lo tanto en el presente trabajo se compara el enfoque de Bandwidth based y Round-robin, para desarrollar el método propuesto, el sistema lo forma un controlador, conmutador Openflow, 4 clientes y 3 servidores y las herramientas HTTperf y OpenLoad y para el entorno SDN el emulador mininet para crear la red virtual y realizar las pruebas de rendimiento de carga, se implementó en SO Ubuntu. En una prueba de 100 solicitudes, en cuanto a rendimiento, se calculó en Kb/s con la herramienta HTTperf, donde la solución basada en cacho de banda obtuvo 65.5 kb/s sobre los 65.1 kb/s de RR y en el tiempo de respuesta se midió con la herramienta OpenLoad, el algoritmo basado en el ancho de bando tuvo mejor rendimiento con 1.025 s sobre el RR que obtuvo 1.035 s En base al resultado la técnica basada en el ancho de banda brinda un mejor tiempo de respuesta y rendimiento a comparación del RR.

Este proyecto de investigación pone énfasis en analizar las diferentes algoritmos balanceadores de carga y su aplicación con la técnica de clustering para poder validar cuál de ellos tiene mejor desempeño y de esta manera poder lograr una reducción de la

caída de los servicios o aplicaciones y que pueda optimizar su disponibilidad, de esta forma se pueda reducir en gran medida la inactividad de un servidor.

Asimismo, con la aplicación de esta solución, las organizaciones tendrán una fortaleza tecnológica lo cual le permitirá brindar siempre un buen servicio a más usuarios, y según su rubro de negocio pueden generar más ingresos, siempre manteniendo una disponibilidad de los servicios que sus servidores implementados.

Por otro lado, desde el punto de vista científico, esta investigación generará conocimiento válido y confiable dentro del área de redes y servidores, como base a investigaciones e implementaciones futuras, ayudando también a empresas e instituciones.

1.2. Formulación del Problema.

¿Cómo mejorar la disponibilidad de los servicios de información de un servidor?

1.3. Hipótesis.

Con la implementación de un clúster de servidores utilizando algoritmos balanceadores de carga, mejorará la disponibilidad de los servicios de información de un Servidor.

1.4. Objetivos.

Objetivo general.

Analizar algoritmos balanceadores de carga para un clúster de servidores para mejorar la disponibilidad de un servidor.

Objetivos específicos.

- Seleccionar algoritmos balanceadores de carga más utilizados y de mejor desempeño.
- Determinar arquitectura de Clúster de servidores para realizar las comparaciones de los algoritmos balanceadores de carga.
- Implementar algoritmos balanceadores en la arquitectura de Clúster de Servidor.
- Realizar las pruebas usando algoritmos balanceadores de carga en un Clúster de Servidores.

1.5. Teorías relacionadas al tema.

1.5.1 Servidor

Un servidor es un ordenador que puede estar implementado en un ambiente físico o virtual, cuya función es brindar uno o varios servicios de diferente tipo a otros ordenadores y/o usuarios ya sea a través de una red interna o externo usando como medio el internet.[24].

Según Marchioni [25], indica que son arquitecturas informáticas que ofrecen un servicio en la red, brindando la información a otros servidores y a clientes, por naturaleza son equipos con mayores características de una PC normal.

De la misma manera precisa que un servidor debe tener varios procesadores, memoria RAM de 16GB a TB a más y un espacio de almacenamiento el cual debe estar limitado a varios discos duros y según estas características pueden ofrecer uno o varios servicios.

1.5.1.1 Hardware interno de un servidor

Almirón [26], define el hardware de servidor a los mismos componentes de un computador básico, pero con mejores características, es decir más especializados para dar un mejor rendimiento a los usuarios de manera continua, lo divide en dos grupos:

Componentes internos

a.- **Mainboard**: Componente que integra a los demás elementos, que puede tener más de un socket, por lo que se convierte en el principal componente de un servidor.

b.- **Microprocesador**: Se caracterizan por tener mejor capacidad y velocidad, teniendo mayor memoria caché, la cual permite realizar procesos con mayor velocidad.

c.- **Memoria**: A diferencia de las memorias convencionales, implementa un buffer que permite detectar y hacer las correcciones de errores (ECC), sin generar sobrecarga en el procesador, también se le conoce como Fully Buffered.

d.- **Disco duro**: Los más utilizados son de tecnología SAS, SATA y SSD.

Figura 1: Disco SATA F1 RAID LG con interfaz SATA.

Fuente: Internet

e.- **Fuente de poder**: Otro componente que se diferencia en los servidores, que por lo general tiene fuentes redundantes, que al fallar una de ellas, la otra continúa trabajando y son de tipo Hot Swap, que no requiere apagar el equipo para retirar una defectuosa.

Tecnología RAID

Es un sistema que se define como un grupo de dos o más discos con la finalidad de obtener mejores beneficios, como mejor velocidad, seguridad, pero dependiendo de su configuración. En 1988 se establecieron niveles de RAID de 1 hasta el 5.

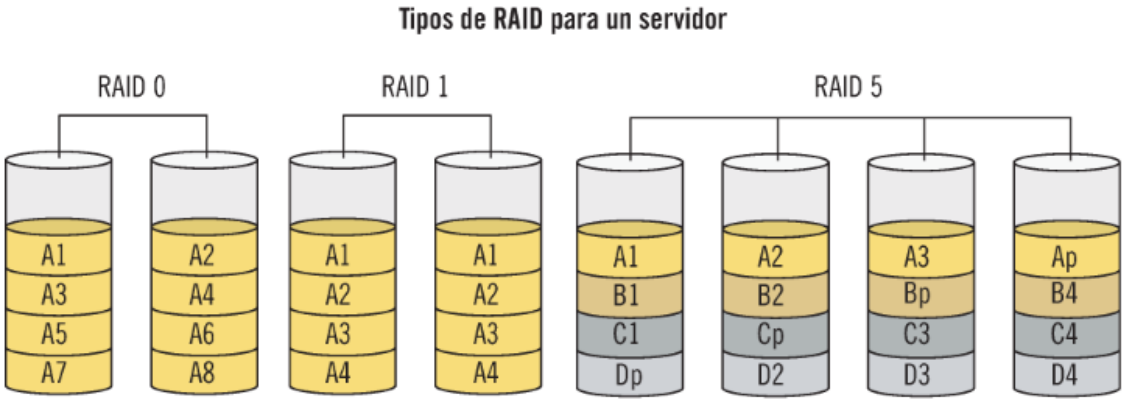


Figura 2: Tipos de RAID para un servidor,

Fuente:[27]

Villada R. [27] menciona que, según la necesidad del usuario, se puede elegir un nivel de RAID, que en sí satisfacen de manera efectiva a uno o dos criterios como seguridad, velocidad, capacidad, costos, tolerancia a fallos.

Según la necesidad del usuario, se puede elegir un nivel de RAID, que en sí satisfacen de manera efectiva a uno o dos criterios como seguridad, velocidad, capacidad, costos, tolerancia a fallos.

- RAID 0. A más discos se obtiene mejor velocidad de transferencia, pero sin tolerancia a fallos, por lo que si un disco de la matriz falla se pierden todos los datos.
- RAID 1. Denominado mirroring que quiere decir “espejado”, cada disco de la matriz funciona como espejo del otro. Permite disponer de una mejor disponibilidad de la información mediante la redundancia, mediante este método permite asegurar la integridad de los datos y su tolerancia a fallos.

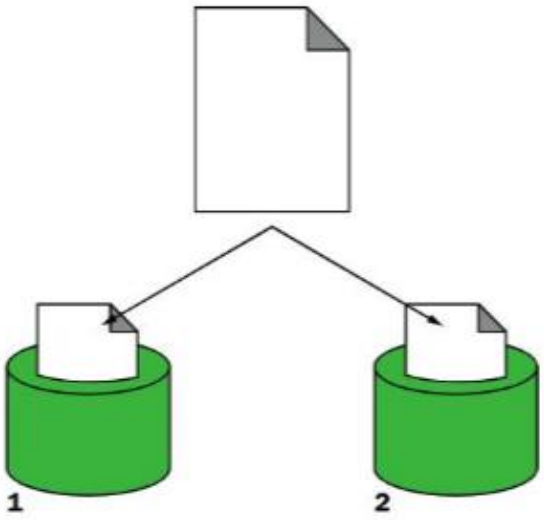


Figura 3: Esquema de un sistema RAID 1

Fuente: Internet

- RAID 0+1. Combinación de RAID 0/1 o RAID 10, con lo cual se logra mejor velocidad y tolerancia a fallos.
- RAID 2. Adapta las técnicas que permite detectar y corregir errores en memoria, está enfocado a aplicaciones que requieren un nivel alto de tasa de transferencia.
- RAID 3. Este método permite disponer de unas altas tasas de transferencia, fiabilidad y disponibilidad, por otro lado, el rendimiento de transacciones es deficiente debido a que los discos que forman la matriz procesan la información al mismo tiempo.
- RAID 4. Esta técnica se utiliza para poder realizar el almacenamiento de archivos de gran tamaño, como aplicaciones de vídeo, sonidos, gráficos.
- RAID 5. Permite una tolerancia a fallos y mejora la capacidad del sistema hasta un 80% de uso del total de discos.

Aplicaciones

Villada R. [27]. Precisa que este componente, están compuestos por las programas o servicios que estarán disponibles para brindar información los usuarios y dependerá del tipo de servidor que se vaya a implementar.

1.5.1.2 Tipos de Servidores

Existen muchos modelos de servidores que pueden ser virtuales o físicos, “son a menudo dedicados, lo que significa que no realizan otras tareas además de sus tareas de servidor”[28], entre los más usados en la actualidad tenemos:

Servidores web

Se le denomina Servidor Web a un software informático que es capaz de procesar aplicaciones del lado del servidor. Realiza el intercambio de datos con el cliente, generando una respuesta en un lenguaje entendible para el usuario, debido a que el

código recibido es compilado, ejecutado e interpretado por un navegador web, “todo este flujo de envió de la información es por medio del navegador y el servidor de comunicarse el uno al otro utilizando el protocolo HTTP”.[28].

Asimismo, este tipo de servidor puede tener un intérprete de lenguaje de programación el cual esté encendido en el Código HTML en las páginas que contiene el sitio web.[29].

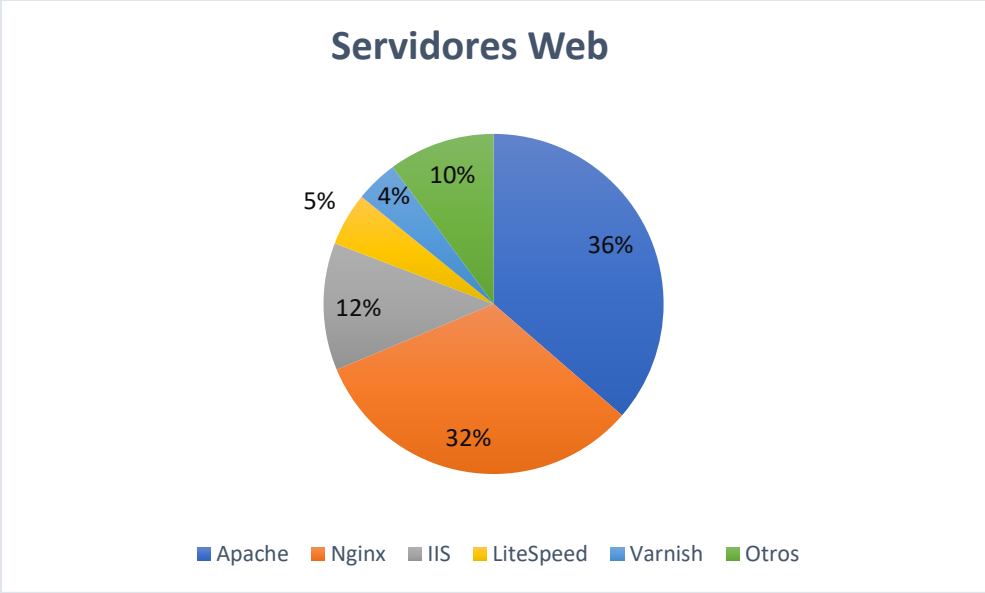


Figura 4. Porcentaje de uso de Tipo de Servidores Web

Fuente: Elaboración propia.

Servidor de Impresión

Según Marchioni [25], este servidor gestiona varias impresoras en red y administra las colas de impresiones según las peticiones de los clientes.

[30], señala que es el que se encarga de gestionar las impresoras que son compartidas por los usuarios de la red.

Servidor de Bases de datos

Estos servidores se caracterizan por gestionar un gran volumen de datos y procesar información, para, manejar toda esta información por lo general se conectan a un Storage.[25].

Servidor de Correo

Tienen la capacidad de poder administrar y direccionar todos los correos de una organización en un solo lugar. Asimismo, trabajan con un almacenamiento, por el alto volumen de información que procesan. [31]

Servidor Proxy

Permiten definir políticas de acceso a internet, por lo general residen en firewalls en donde se configuran estas reglas para permitir la navegación por determinadas páginas.

1.5.2 Capacidad de carga de un servidor

Por naturaleza los servidores tienen una capacidad de carga según características, para atender un número determinado de peticiones simultáneas

Para afrontar las sobrecargas, se pueden aplicar un conjunto de técnicas, dentro las cuales tenemos a los balanceadores de carga, la cual sería una alternativa de solución para distribuir el tráfico en una arquitectura de varios servidores. Los balanceadores de carga además ofrecen otras funciones que se combinan para asegurar la disponibilidad de un servidor.[32].

1.5.3 Clúster de balanceo de carga

Es un sistema que está compuesto por uno o más ordenadores, a quienes se les denomina nodos, que actúan como front-end del clúster, quienes se encargan de dividir las peticiones de servicio que reciben de parte de los clientes, a otros servidores que forman el back-end, con esta tecnología se puede ampliar su capacidad agregando más

ordenadores y ante la caída de uno de los ordenadores del clúster el servicio que ofrece seguirá funcionando.

1.5.4 Clúster de servidor

Borges [33]. Un grupo o clúster de servidores, consiste en la interconexión de varios sistemas informáticos (servidores) con el fin de funcionen con un único servidor. Al hablar de «unir» hacemos referencia que estos sistemas informáticos hacen uso de los mismos recursos de tipo físicos y lógicos, logrando funcionar, como un sistema integrado. Esta unión de componentes se realiza según las necesidades, aunque el factor principal es mejorar la velocidad y por otro lado lograr una alta disponibilidad ante fallos

Fernandez [34]. menciona que al crear un clúster de servidor no es solo conectarlos, si no también es necesario de un software que permita controlar y optimizar el funcionamiento, asimismo precisa que en el tiempo han evolucionado y son aplicados en diversos ámbitos como servidores web, tareas de cómputo, en bases de datos y en el comercio electrónico.

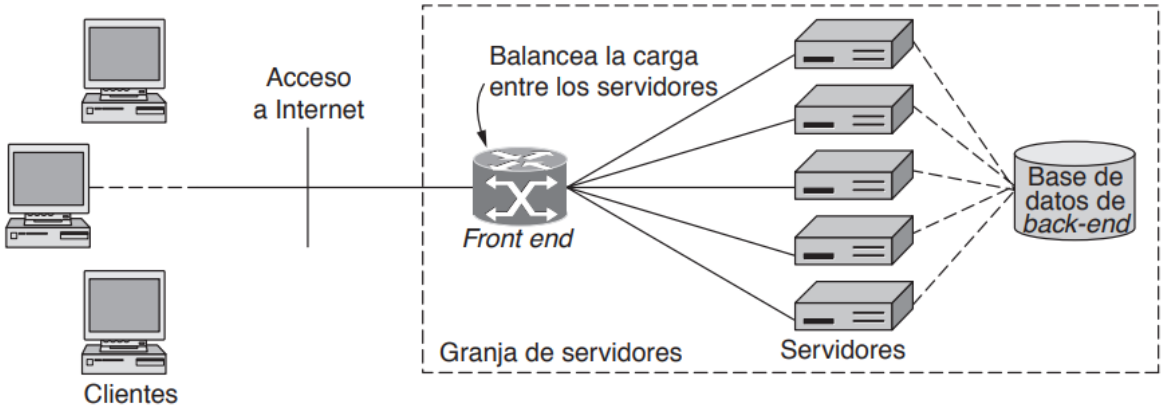


Figura 5. Esquema de un clúster de servidores.

Fuente: Imagen de internet.

1.5.4.1 Funcionamiento de un clúster

Los clústeres de servidores funcionan bien para aplicaciones que no se pueden asignar entre varios servidores. Se dice que cada servidor incluido en un clúster de servidores posee y controla sus dispositivos locales. Cada servidor también mantiene una copia del sistema operativo particular, sus aplicaciones y servicios diseñados por el clúster. Cada clúster tiene dispositivos comunes como discos que se encuentran en el suministro de disco común. Un medio de conexión que puede proporcionar acceso a estos discos, etc. también está poseído y controlado por un servidor a la vez.

En un clúster de servidores, solo funcionará un nodo a la vez. La razón es que cada nodo recibe sus actualizaciones por separado. Los otros nodos generalmente se mantienen en modo de espera. Si algún nodo activo falla, los otros nodos en espera se harán cargo de inmediato. Este proceso es posible porque todos los nodos de un clúster están conectados a un sistema de almacenamiento compartido. Este sistema de almacenamiento compartido utilizado por los clústeres de servidores se denomina quórum. Se dice que es la distribución de la base de datos del clúster de servidores. El recurso de quórum generalmente contiene datos como información sobre la configuración del clúster y mantiene un registro de los cambios actuales realizados en esa configuración.

Características

- Son escalables, en la cual tienen la capacidad de crecer para aumentar las prestaciones del servicio.
- Los clústeres integran un software de gestión para realiza el monitoreo y gestionar los recursos de los nodos que lo forman

1.5.4.2 Clasificación de Clúster de servidor

Según Guerrero [35], un clúster ha sido construido de manera minuciosa, lo cual le permite garantizar factores óptimos de calidad, como la integridad, eficiencia y facilidad de mantenimiento.

- **Clúster de alto rendimiento** (High Performance Computing Clúster): “Su aplicación es en ambientes de red que realizan procesos o tareas que implican un alto nivel de procesamiento de la información, por lo que hacen uso de elevado de CPU, memoria o velocidad de lectura/escritura en disco”. [36]
- **Clúster de alta disponibilidad** (High Availability Computing Cluster): Ofrecen una mejor garantía ante fallos de red, hardware o software, lo que permite que una aplicación siempre esté disponible para el acceso de los usuarios las 24 hs.
- **Clúster de alta eficiencia** (High Throughput Computing Cluster): Utilizado cuando se necesita de un eficiente procesamiento de datos, procesando grandes cantidades de peticiones en el menor tiempo posible.

1.5.5 Alta disponibilidad

El funcionamiento de clúster de servidor, permanece transparente al cliente con independencia del estado de funcionamiento de los servidores internos, en la cual los clientes pueden disponer de los datos en todo momento.[37].

Martinez [38], lo define como un servicio que está el mayor tiempo posible funcionando, de preferencia esté disponible y accesible las 24 horas del día, 7 días de la semana y 365 días del año, de esta manera ofrece un 100% de disponibilidad. Asu indica que es este caso es imposible de implementar debido a su dependencia de hardware y software que en algún momento pueden llegar a presentar fallas, por otro lado, también genera gastos elevados. Finalmente, precisa que lo ideal es encontrar un equilibrio entre ambos el costo y la alta disponibilidad.

Según Jimenez [39], La Alta Disponibilidad (HA), “permite la duplicación de hardware. Lo cual nos va facilitar reducir ciertos problemas que puedan afectar al servicio que ofrece una organización. Si bien los servidores puedan tener caídas en su servicio y que, en definitiva, no se tenga el acceso a la información que almacenan. Al hacer uso de HA, es que en caso de que falle un servidor, no afectar al servicio. Debido a que automáticamente se sustituye por otro. Esto se logra al haber duplicación de hardware, como se ha mencionado”, anteriormente.

Ventajas y desventajas

Disponibilidad. El clúster sigue operando ante un fallo de alguno de sus ordenadores que lo conforman, distribuyendo las peticiones de manera paralela.

Flexibilidad. Los balanceadores de carga son independientes de la arquitectura de hardware.

Costos. La inversión en diseño y montaje son menores en comparación a otras alternativas de solución.

Escalabilidad. Capacidad de poder asumir volúmenes de trabajo el cual va en aumento, sin que afecte al rendimiento.

Expansibilidad. Capacidad de aumentar sus capacidades de arquitectura y

Incremento de número de transacciones y velocidad de procesamiento.

Desventajas

Las empresas, prefieren utilizar la arquitectura cliente/servidor tradicional, debido al espacio físico y algunos problemas que no se presentan en un modelo convencional.

1.5.6 Componentes de un clúster

Asimane [40], menciona a los siguientes componentes que forman un clúster.

Nodos

Representan un grupo de equipos que pertenecen a un clúster, que contiene recursos de unidades físicas o lógicas.

Sistema operativo

Es un programa de computadora que permite la gestión de sus recursos, y deben ser de un entorno multiproceso, multiusuario a fin de que los usuarios puedan conectarse y realizar procesos. Estos entornos pueden ser Unix, Linux, Windows o Solaris. Cuanto más fácil sea el uso se reducen problemas en su administración.

Conexión de red

Un clúster está implementado en una red para comunicarse con los usuarios y una red diferente para que los servidores se comuniquen entre ellos.

Middleware

Es un aplicativo que actúa de manera independiente con el Sistema Operativo y las aplicaciones, ofreciéndole al usuario de estar haciendo uso de un computador de mucha capacidad de procesamiento.

A su vez ofrece una interfaz denominada SSI (Single System Imagen), que facilita herramientas para el mantenimiento de procesos pesados como balanceo de carga, tolerancia a fallos entre otros, a su vez permite la escalabilidad del clúster.

Almacenamiento

En la mayoría de aplicaciones instaladas en el servidor, los equipos deben contar con un espacio de almacenamiento que sea compartido y accesible desde los nodos del clúster.

Protocolos de conmutación

Es una técnica que permite hacer uso eficiente de los enlaces que se aplica para los nodos que forman el clúster.

Aplicaciones

Son las que están instaladas y disponibles en el clúster para los usuarios externos.

1.5.7 Balanceadores de carga

Es una técnica que se utiliza para compartir los procesos entre varios ordenadores, disco u otros recursos, esta técnica se mantiene mediante un algoritmo que divide de una forma más equitativa las peticiones a fin de evitar una sobrecarga o cuello de botella, según [38].

En la actualidad se hacen uso en varios escenarios como servicios de DNS, Web, Correo, Bases de datos, etc. [41].

Ainoa [42], los define como un componente de hardware o software que se ubica al delante de un grupo de servidores que atienden una aplicación, asignando las peticiones que llegan de los usuarios a los servidores, mediante el uso de algún algoritmo.

Wacker [43]. Menciona que existen varias estrategias para optimizar el balanceo de carga. Como el "Round Robin" (que distribuye las solicitudes que envían los usuarios desde el Internet entre el conjunto de nodos o servidores que están disponibles ofreciendo dicho servicio), a su vez los servidores que atienden las solicitudes, recopilan los datos en tiempo real, sobre la capacidad operativa de los nodos y la hacen uso para tener un mejor enrutamiento de dichas peticiones a los servidores que tengan una mejor disponibilidad de atender los servicios. Los métodos de balanceo de carga, si bien pueden ser soluciones de tipo hardware, mediante routers y switches que incluyen software de distribución de carga ya establecidos, Asu vez se puede configurar en el back end de los servidores.

El uso de estos equipos radica en que permiten repartir la carga y excluir aquellas conexiones de destino que se saturan o caídas en un momento determinado, estas son direccionadas a otro servidor[44].

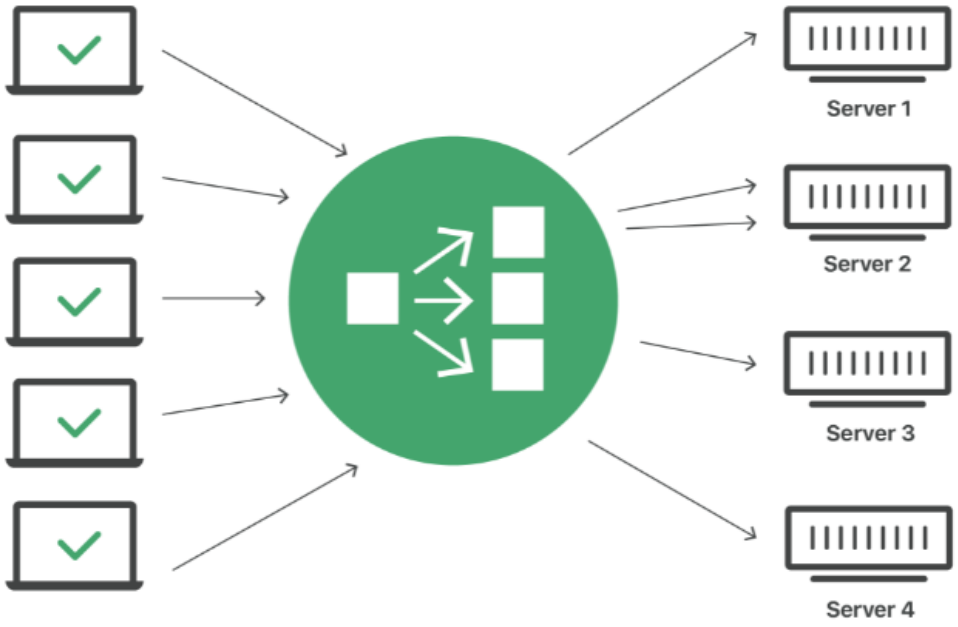


Figura 6. Un esquema de red haciendo uno de una técnica de balanceo de carga**.**

Fuente**:** [45]

Córcoles [37], Los balanceadores de carga tienen varios uso y beneficios, de los cuales mencionamos algunos principales objetivos de un balanceador de carga.

Escalabilidad

El incremento de la potencia se obtiene adicionando más elementos o cursos al clúster, en un proceso lineal, sin embargo, en algunas situaciones mucho dependerá de la aplicación.

Eficiencia

Un clúster permite administrar ante una variedad de peticiones de forma que se contemple una carga equilibrada entre cada nodo, estos son de mucha importancia cuando se hacen uso de equipos de distintas potencias y características de hardware.

Disponibilidad

Para lograr la disponibilidad, la información es replicada entre los otros discos que conforman el clúster, de manera que ante la posible falla de uno de ellos es transparente a los clientes.

Transparencia

Si bien, al existir un solo clúster, hace que un solo servidor sea inapreciable desde el lado de los usuarios que hacen uso de los servicios del servidor.

1.5.8. Tipo de algoritmos

Existen varias técnicas de balanceo de carga, estáticos y dinámicos, que se pueden implementar en entornos homogéneos o heterogéneos.

Tabla 1. Características y diferencias de los algoritmos de balanceo de carga.

Algoritmo	Método de solicitud	Naturaleza	Observación	Clúster adecuado [b]
Round Robin	La petición se envia al servidor en forma aleatoria.	Dinámico	Fácil de configurar desplegar y ampliamente algoritmo usado	. Homogéneo
Least Connections	La petición se envía al servidor con el menor	Dinámico	Evita sobrecargando un servidor verificando el número de	Homogéneo

	número de conexiones		conexiones de servidor	
Weighted Round Robin	Cada servidor se utiliza, a su vez, por peso.	Estático	Puede enviar más solicitudes a servidores más capaces y cargados	Homogéneo & Heterogéneo.
Source IP Hash	La dirección IP de origen se dividirá por el número total de servidores en funcionamiento para decidir qué servidor recibe la solicitud.	Estático	Los usuarios se conectan a una sesión aún activa después de la desconexión y reconexión. Aumentará rendimiento.	. Homogéneo & Heterogéneo
Least Response Time	La petición se envía al servidor con el menor tiempo de respuesta.	Dinámico	Considere tanto el la capacidad del servidor, tiempo de respuesta y el número de conexiones actuales para evitar la sobrecarga y chocar.	Homogéneo

Least Bandwidth	Cada servidor se utiliza, a su vez, según el ancho de banda de la red.	Estático	Puede enviar más solicitudes a más capaz y la red ancho de banda cargado servidores.	Homogéneo & Heterogéneo
Throttled	La solicitud se acepta si se encuentra una coincidencia en la tabla; de lo contrario, se devuelve y la solicitud se pone en cola.	Dinámico	intenta distribuir la carga de manera uniforme entre las máquinas virtuales	Homogéneo

Nota: Tomado de [17] , Chang (2020, P. 24). [b]Chiang, Cheng1, Liu, Chiang1 (2020, P. 10)

1.5.8.1. Source IP hash

En esta técnica, la dirección IP de origen de las peticiones de un cliente se calcula utilizando el algoritmo de hash, para obtener una clave de hash única, y todos los servidores backend están numerados. La clave generada se asigna al cliente a un servidor en particular. Esto permite enrutar las solicitudes de diferentes clientes en función de las direcciones IP de origen y garantiza que un cliente se dirija al mismo servidor que estaba utilizando anteriormente.

En la siguiente figura, se observa cómo se distribuyen las peticiones utilizando el algoritmo hash de IP de origen. En la que dos servidores backend están disponibles y tienen el mismo peso. Si el servidor backend 01 ha procesado una solicitud desde la dirección IP A, el balanceador de carga enviará las nuevas solicitudes desde la dirección IP A al servidor backend 01.

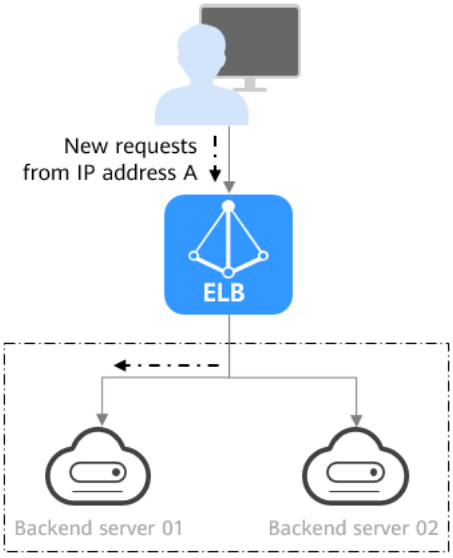


Figura 7. Distribución del tráfico del algoritmo hash de IP de origen. Fuente: [46]

1.5.8.2. Weighted least connections

Esta técnica deriva del algoritmo least connections, que utiliza la cantidad de conexiones activas a cada servidor backend, para tomar su decisión de equilibrio de carga. Asimismo, del número de conexiones, a cada servidor en función a su capacidad se le asigna un peso. Estas solicitudes se enrutan al servidor con la menor proporción de conexiones a peso. Este algoritmo mayormente se utiliza para conexiones persistentes a bases de datos.

Los servidores con un valor de peso más alto recibirán un mayor porcentaje de conexiones activas en cualquier momento, es decir, introduce un "peso" que se basa en las especificaciones de cada servidor. (Gurasis & Kamalpreet, 2018)

En la siguiente figura se observa cómo se distribuyen las solicitudes mediante el algoritmo de WLC. Dos servidores backend tienen el mismo peso, se han establecido 100 conexiones con el servidor backend 01 y se han conectado 50 conexiones con el servidor backend 02. Las nuevas solicitudes se enrutan preferentemente al servidor backend 02.

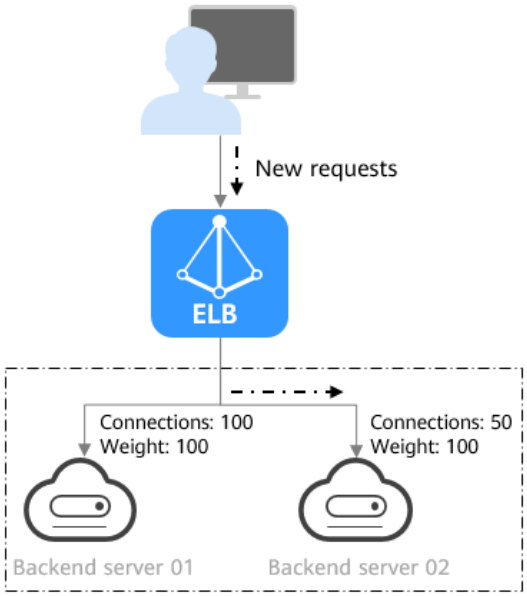


Figura 8. Distribución del tráfico utilizando el algoritmo de WLC.
Fuente: [47]

1.5.8.3. Weighted round robin

Las peticiones se distribuyen entre los servidores backend en secuencia según sus pesos. Donde los servidores backend con pesos más altos reciben proporcionalmente más solicitudes, por otro lado, los servidores con ponderaciones iguales reciben la misma cantidad de solicitudes. Este algoritmo se utiliza normalmente para conexiones cortas, como conexiones HTTP.

Como se observa en la siguiente figura, cómo se distribuyen las solicitudes mediante el algoritmo WRR. Dos servidores backend están disponibles y tienen el mismo peso, y cada servidor recibe la misma proporción de solicitudes.

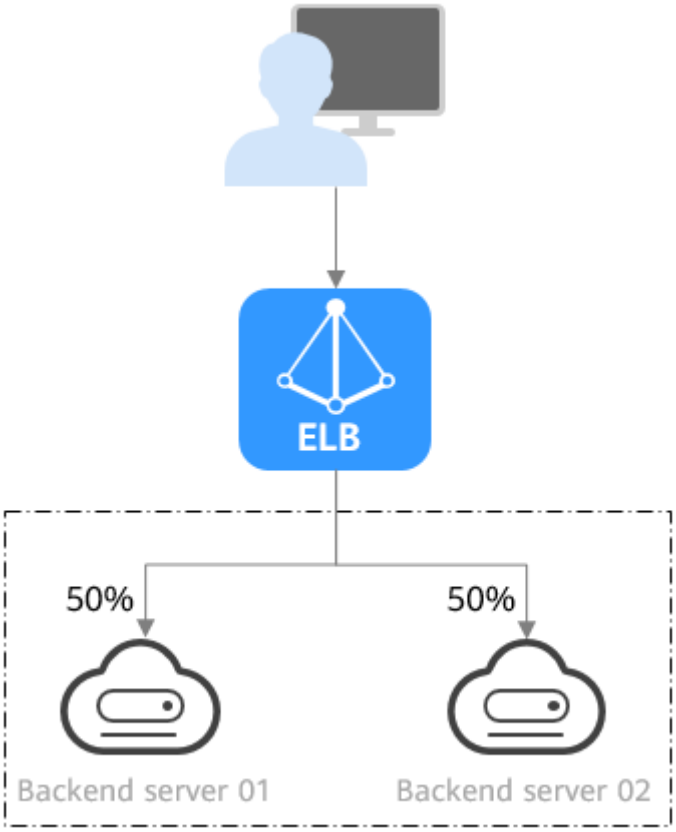


Figura 9. Distribución del tráfico utilizando el algoritmo WRR.

Fuente: [47]

1.5.8.4. Connection ID

El ID de conexión en el paquete se calcula utilizando el algoritmo hash consistente para obtener un valor específico, y los servidores backend están numerados. El valor generado determina a qué servidor backend se enrutan las solicitudes. Esto permite enrutar solicitudes con diferentes ID de conexión y garantiza que las solicitudes con el mismo ID de conexión se enrutan al mismo servidor backend. Este algoritmo se aplica a solicitudes de aplicaciones que utilizan QUIC como protocolo.

Un ejemplo de su funcionalidad se observa en la siguiente figura, donde se distribuyen las solicitudes mediante el algoritmo de ID de conexión. Dos servidores

backend están con disponibilidad y tienen el mismo peso. Si el servidor backend 01 ha procesado una solicitud del cliente A, el equilibrador de carga enruta las nuevas solicitudes del cliente A al servidor backend 01

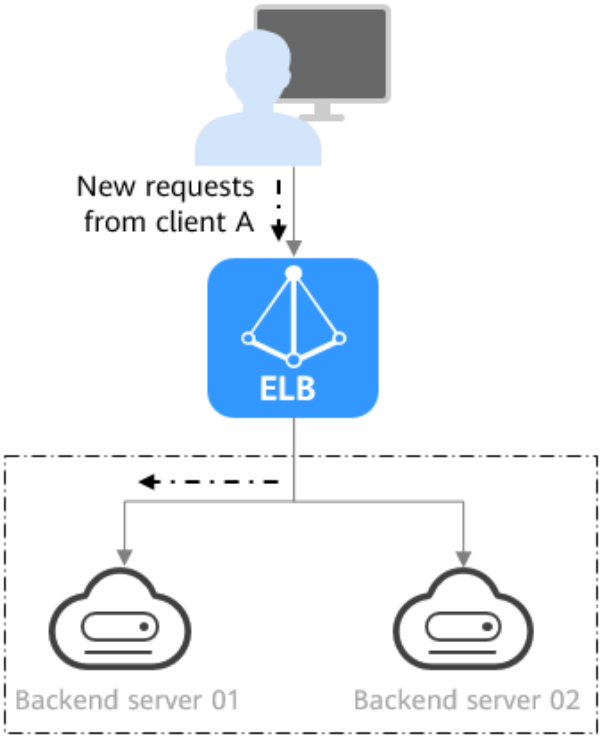


Figura 10. Distribución del tráfico mediante el algoritmo Connection ID.

Fuente: [47]

1.5.3.5. Round-Robin

Esta técnica implica un método simple de equilibrar la carga de los servidores, que funciona mejor cuando todos los servidores backend tienen una capacidad similar y la carga de procesamiento requerida por cada solicitud no varía significativamente.

Se prioriza que los servidores sean idénticos para ofrecer los mismos servicios. Cada uno está configurado para usar el mismo nombre de dominio de Internet con direcciones IP únicas.

Ejemplo de Configuración de lado Fronted.

```
frontend rrlb
  bind *:80
  option http-server-close
  option forwardfor
  default_backend app-rr
```

Ejemplo de Configuración de lado Backend

```
backend app-rr
  balance roundrobin
  option httpchk HEAD / HTTP/1.1\r\nHost:\ localhost
  server web-server-1 rrserver01:80 check
  server web-server-2 rrserver01:80 check
  server web-server-2 rrserver01:80 check
```

El algoritmo round robin depende de un sistema de rotación o selecciona de forma rotativa los servidores. Como se muestra en la Figura 5. Este algoritmo es, en general, beneficioso mientras funciona con servidores de idéntica importancia ya que funciona mejor cuando los recursos de las máquinas son los mismos.

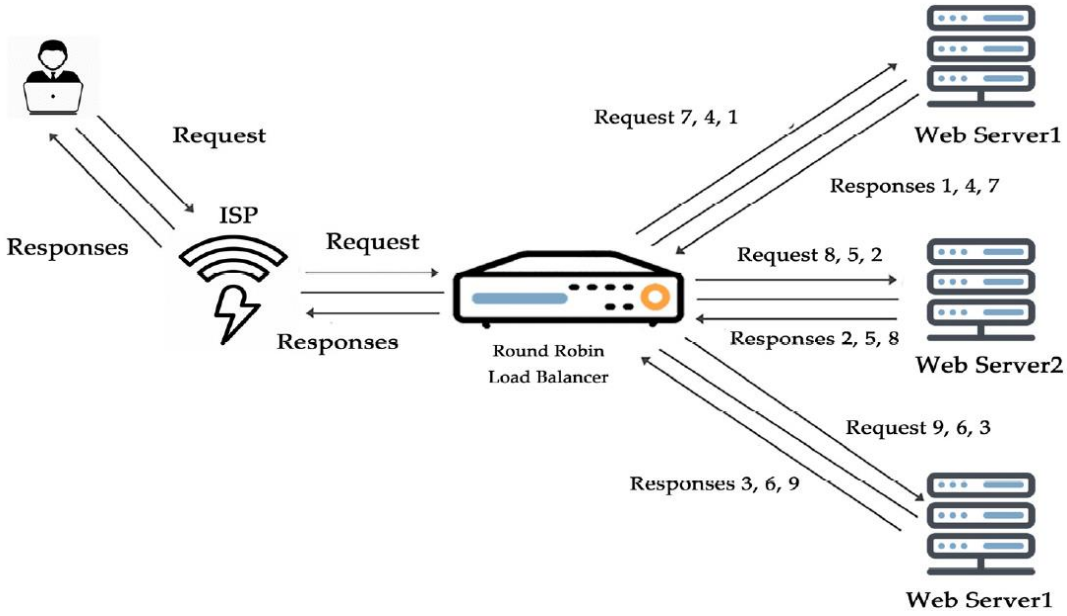


Figura 11. Funcionamiento del algoritmo Round Robin.

Fuente:[17]

1.5.8.6. Least Connections:

Esta técnica permite dirigir las solicitudes al servidor que disponga de la menor cantidad de sesiones activas, es decir se distribuyen las solicitudes de los clientes entre varios servidores según la cantidad de conexiones existentes en el servidor. En cambio, las conexiones nuevas se envían al servidor con menos conexiones.

Esta funcionalidad, se basa en que hace seguimiento en tiempo real de la carga de los servidores reales a mediante la tabla IPVS. Lo cual hace necesario su uso cuando hay un alto grado de variaciones en la carga de datos. Si un grupo de servidores tiene diferentes capacidades, la programación least-connection es la mejor opción.[48]

Configuración de lado Fronted.

```
frontend lclb
  bind *.80
  option http-server-close
  option forwardfor
  default_backend app-lc
```

Configuración de lado Backend

```
backend app-lc
  balance leastconn
  option httpchk HEAD / HTTP/1.1\r\nHost:\ localhost
  server web-server-1 lcserver01:80 check
  server web-server-2 lcserver01:80 check
  server web-server-2 lcserver01:80 check
```

1.5.8.7. Least Bandwidth

Este algoritmo de ancho de banda mínimo procesa las solicitudes en megabits (Mbps) por segundo, enviando las peticiones de los usuarios al servidor con el menor Mbps de visitantes como se observa en la Figura 7. La técnica de balanceo de menor ancho de banda es muy útil cuando todas las máquinas en la infraestructura tienen un ancho de banda de red diferente. Requiere un ancho de banda de red aproximado, lo cual es difícil de hacer en redes donde el tamaño del paquete de datos varía y el ancho de banda de la red puede estar agotado.[17].

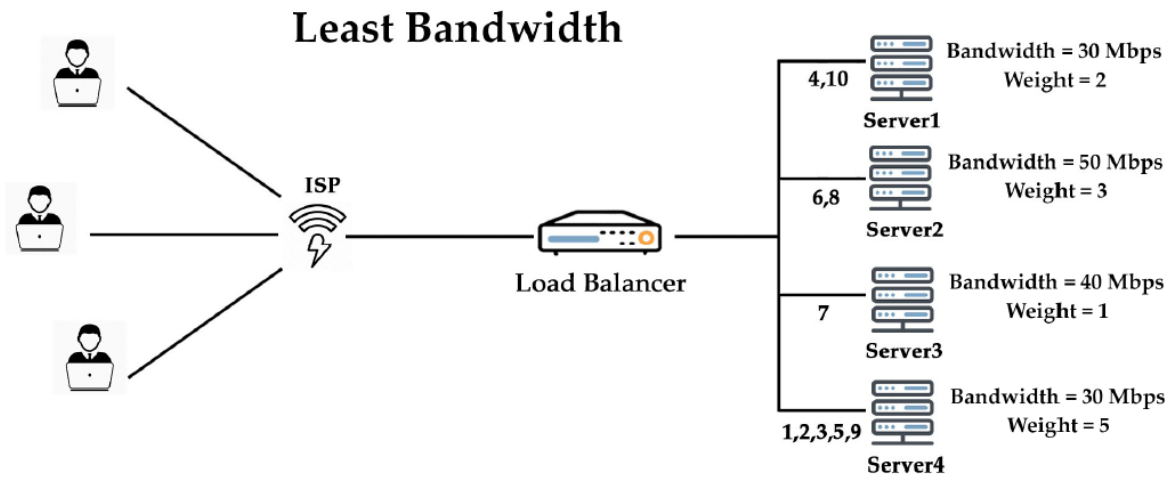


Figura 12. Funcionamiento del algoritmo Least Bandwidth.

Fuente:[17]

1.5.8.9. Weighted Least Connection

Este algoritmo se aplica a los servidores de aplicaciones, no a los servidores de autenticación o autorización, y requiere la función de optimización de aplicaciones.[49], mantiene el número de conexiones activas a cada servidor backend, se asigna un peso a cada servidor backend en función de su capacidad de procesamiento. Es utilizado mayormente para conexiones persistentes, como conexiones de bases de dato.[50].

Los servidores con un valor de peso más alto recibirán un mayor porcentaje de conexiones activas en cualquier momento, es decir, introduce un "peso" que se basa en las especificaciones de cada servidor.[51]

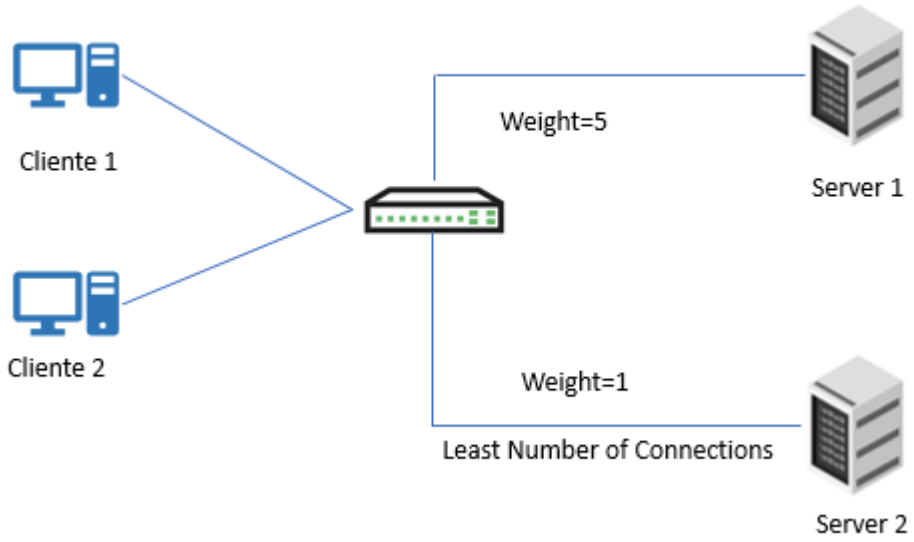


Figura 13: Funcionamiento del algoritmo Weighted Least Connection.

Fuente:[17]

1.5.9 Funcionamiento

El equilibrio de carga de red permite asegurar que el conjunto de servidores tenga una carga de trabajo uniforme.[52]

Caballero & Clavero [41], ilustran este funcionamiento en un escenario Web de una gran empresa: este sistema se compone de un clúster para el procesamiento de la página web y un balanceador, el acceso para los usuarios se realiza mediante una dirección IP y será el balanceador de carga el que decide el reparto de las conexiones entre los distintos nodos del clúster, con el fin de que se realice su procesamiento.

Esta capacidad de dividir la carga de trabajo se realiza mediante métodos basados en algoritmos como Round-Robin, Weighted Round-Robin, LeastConnection y Weighted LeastConnection entre otros.

Teniendo como ventajas, un alto rendimiento en el sistema a un costo menor, totalmente transparente para el usuario y sobre todo que evita una sobrecarga de procesos en los nodos del clúster.

1.5.9 Tipos de balanceadores.

Fernandez [34], precisa que se clasifican en los siguientes:

- **De tipo hardware:** El cual es un servidor exclusivo que tiene un tipo de sistema operativo, y un software para realizar la gestión de balanceador de carga. Estos tipos de servidores integran a los servidores web mediante la tecnología de Plug and Play, lo que permite de forma automática detectar cuando son conectados, funcionan con mínimas configuraciones.
- **De tipo switch:** El balanceador de carga de esta característica requiere de un switch Layer 2 o 3 para hacer uso de la técnica de balanceo. Por lo que no es necesario contar con un dispositivo adicional o intermediario entre el switch y el servidor web.
- **Basado en software:** Se caracteriza por no tener la necesidad de hacer cambios en su característica de conectividad de red. Se puede realizar la instalación del software en los propios servidores web. Sin embargo, se tiene la alternativa de hacer uso de un servidor exclusivo que le permita cumplir el rol de un balanceador de carga.

1.5.10. HAProxy

Es un software de equilibrio de carga TCP / HTTP de código abierto y una solución de proxy para aplicaciones basadas en TCP y HTTP. El modo de operación de HAProxy hace que su integración en arquitecturas existentes su uso es práctico y sin riesgo además es adecuado para sitios web de alto tráfico y funciona con varios de los más visitados del mundo.[53]

HAProxy (siglas de High Availability Proxy) es un equilibrador de carga de código abierto que se usa a menudo con Apache.

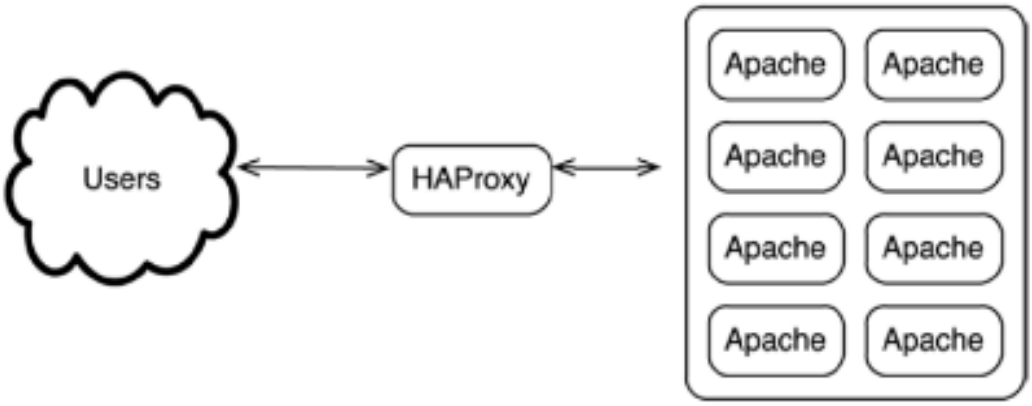


Figura 14: Funcionalidad de Acceso usuario servidor HaProxy.

Fuente: Internet

Características más importantes

- Es compatible con un amplio universo de algoritmos de balanceo de carga.
- Contiene listados de control de acceso, para mejorar la gestión de controlar el flujo de contenido, mediante reglas establecidas.
- Soporte SSL nativo: Permite mejorar la seguridad del contenido.
- Ofrece protección ante ataques de denegación de servicio, a nivel de servidor proxy mediante la protección DDoS.
- Soporta el nuevo estándar de direccionamiento en IPv6
- Compresión HTTP/1.1: Para minimizar el consumo de ancho de banda

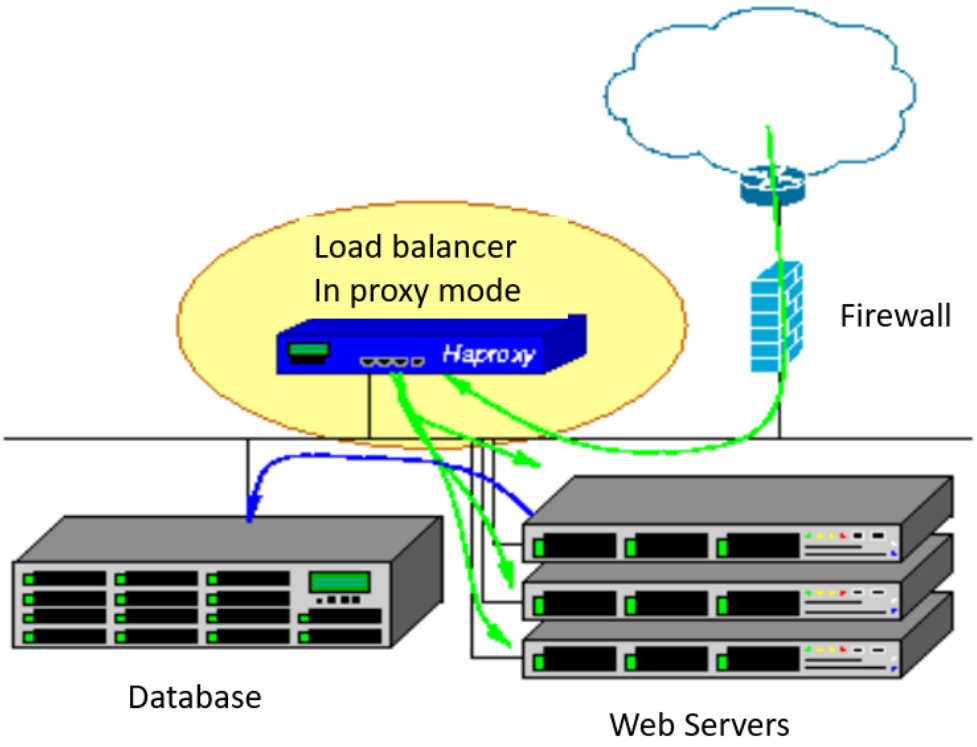


Figura 15: Haproxy, haciendo uso de Balanceador de carga Aproxy.

Fuente: [54]

1.5.11. Nginx

Villada [27], lo define como un Servidor HTTP y proxy inverso de alto rendimiento, a su vez funciona para los protocolos de correo como IMAP/POP3/SMTP; es conocido por ser estable, características, configuración simple, y no consume demasiados recursos, lo que le hace recomendable para sitios web que necesitan velocidad y eficiencia.

Del mismo modo se caracteriza por que gestiona solicitudes basado en eventos, que le permite asegurar el funcionamiento óptimo ante sobrecarga de peticiones.

Borges [55] nos menciona que Nginx fue creado con la finalidad de ser un servidor web escalable, eficiente y rápido, siendo una característica que permite que los sitios web más usado del mundo lo utilicen para mejorar su infraestructuras de servidores HTTP, de mailing, balanceadores de carga, servidores proxy, etc.

Funcionamiento

A diferencia de otras aplicaciones de servidores web como Apache. Su arquitectura se basa de forma asincrónica, ejecutándose por eventos, por lo que toda petición que proviene de un navegador web se gestiona aplicando un solo thread o hilo. Lo que le permite no consumir muchos recursos, en cuanto a memoria, de esta manera gestándose como una plataforma para el servicio de peticiones web rápida y liviana.

Este tipo de servidor web funciona en base a procesos maestros, los cuales permiten controlar procesos de trabajo, que se logran dar respuesta de las peticiones que envían los clientes mediante el uso de navegadores web. [55].

Características

Estas características son importantes a tener en cuenta al momento de elegir su uso en proyectos de infraestructura tecnológica.

- Permite tener como un servidor proxy inverso y servidor de caché
- Facilita el uso de Servidor de archivos estáticos y dinámicos
- Soporta la parte de autenticación HTTP y las técnicas de Balanceo de carga
- Se obtiene una alta disponibilidad y Escalabilidad
- Se adapta a los protocolos IPv4 e IPv6 y brindando soporte HTTP, HTTP2 y HTTPS con certificados de seguridad (SSL)
- Permite implementar hosts virtuales.
- Se implementan como Proxy para POP3, SMTP e IMAP con soporte SSL

II. MATERIALES Y MÉTODO

2.1. Tipo y Diseño de Investigación.

2.1.1. Tipo de investigación.

En el desarrollo de esta investigación se aplicará una investigación tecnológica y aplicada. Baena [56] Define que la investigación aplicada tiene como propósito el estudio de un realidad problemática destinada a la acción, la cual aporta nuevos hechos y enfocan sus esfuerzos en la solución de las necesidades que es necesario para la sociedad y el hombre; asimismo [57]., indica que comprende de un conjunto de actividades que tienen por propósito el encontrar o aplicar nuevos conocimientos científicos, para la solución de problemas dirigido a conseguir mejoras en los procesos o bien incremento en la calidad y productividad el objeto en estudio.

En cuanto al tipo de investigación tecnología, [58] Menciona que su propósito es de dar solución a problemas o situaciones del conocimiento científico y de esta manera elaborar procesos en relacionado a estudios ya ejecutados, lo cual posee un carácter práctico y determinado en la que se ha planteado la necesidad o problemática y en base a de ello buscar resolver teniendo en cuenta las teorías existentes y el conocimiento empírico.

2.1.2. Diseño de la investigación

Es cuantitativa, porque se va contrastar la hipótesis en un enfoque probabilístico orientado al resultado y de ser aceptadas y demostradas, poder formular teorías generales.[59].

Esta investigación es cuasiexperimental, porque tiene como objetivo analizar algoritmos balanceadores de carga para un clúster de servidor para mejorar la disponibilidad de un servidor, según [60], hacen mención que mediante este tipo de

diseño nos permite acercarnos a los resultados de una investigación experimental en la que no es posible poder el tener el control ni la manipulación absoluta de las variables.

2.2. Variables, Operacionalización.

Tabla 2: Operacionalización de variables

Variable de estudio	Definición conceptual	Definición operacional	Dimensiones	Indicadores	Ítems	Instrumentos	Valores finales	Tipo de variable	Escala de medición.
V. I. Clúster de servidores utilizando algoritmos balanceadores de carga	Grupo de servidores que administran los recursos mediante el balanceo o distribución de peticiones entre sus nodos.	Realizar pruebas para identificar el algoritmo que tenga un mejor desempeño en cuanto a tiempo de repuesta, consumo de recursos y mayor velocidad de transferencia datos, medidos en el clúster de servidor mediante la ficha digital.	Tiempo de respuesta Consumo de Recursos Ancho de Banda	Tiempo Procesamiento Número de Usuarios Consumo de RAM y CPU Usuarios conectados Velocidad Transferencia en kbps		Ficha digital de observación Software de gestión de red	Menor tiempo de respuesta Menor consumo de RAM y uso de HD Mayor velocidad de transferencia de datos en kbps	Categórica	Ordinal
V. D. Mejorar la disponibilidad de un Servidor.	Estado en el que un servidor está disponible ante las peticiones de los usuarios.	Realizar pruebas para identificar el algoritmo que permita un mejor performance en cuanto a, rendimiento y disponibilidad de un servidor, medidos a través de la ficha digital.	Rendimiento Disponibilidad	#bits procesados por segundo Unidad de Tiempo de procesamiento Tiempo de Disponibilidad		Ficha digital de observación Software de gestión de red	Mayor procesamiento de #bit'ps Mayor tiempo de respuesta	Categórica	Ordinal

Nota: Tomado de [a] (Joshi & Gupta, 2019, p. 147)[61] [b] Hsien-Yi L.(2018, p. 102) [11].

2.3. Población de estudio, muestra y criterios de selección.

2.3.1. Población

La población está conformada por 16 algoritmos balanceadores de carga, los cuales se encuentran en Anexo N° 02, los que fueron identificados en la revisión sistemática de la literatura.

2.3.2. Muestra

La muestra ha sido determinada mediante las técnicas de no probabilística por conveniencia y está compuesta por 04 algoritmos balanceadores de carga más utilizados y de mejor rendimiento que son los siguiente:

Tabla 3: Muestra del Estudio

Algoritmos	Identificador
Weighted Round Robin	(WRR)
Least Conection	(LC)
Round Robin	(RR)
Consistent Hashing	(DCH)

2.4. Técnicas e instrumentos de recolección de datos, validez y confiabilidad.

2.4.1. Técnicas de Observación

Esta técnica tiene como principal objetivo, que el adecuado criterio poder percibir la realidad de la ejecución de un método en búsqueda de un resultado que demuestre la verdad o falsedad de la hipótesis, y con ello contribuir a la investigación del tema relacionado.

2.4.1.1 Instrumentos

a. Reporte de consumo procesos.

Esta herramienta permitirá conocer datos de consumo de los componentes principales del servidor como CPU y Memoria.

b. Ficha digital de observación.

Es un instrumento en la cual se anotarán los resultados durante las pruebas a ejecutar de los algoritmos, la cual permite tener una forma ordenadas los datos obtenidos para realizar su procesamiento según los indicadores, estas fichas se encuentran en el Anexo 05.

2.5. Procedimiento de análisis de datos.

Para el procedimiento de recolección de datos se tendrá.

2.5.1. Tiempo de respuesta

Indicador que muestra la cantidad de tiempo para procesar la solicitud por parte de los servidores cuando el usuario envía una solicitud, su fórmula es.

$$TR = \frac{TPT}{TNU}$$

Donde:

TR: Tiempo de respuesta

TPT: Corresponde al tiempo total de procesamiento

TNU: Corresponde al número total de usuarios

2.5.2. Consumo de servidor

Indicador que muestra el uso de los componentes internos del servidor, como el CPU, Memoria RAM, que da en respuesta a los procesos de las peticiones que atiende y según este valor de consumo permita al balanceador de carga distribuir las peticiones a servidores que tenga menor carga de uso, su fórmula es:

Loading= cpu utilization +mem usage

Donde:

Loading: consumo

cpu utilization: uso de CPU

mem usage: Uso de memoria

2.5.3. Ancho de banda

Indicador que permite conocer como la cantidad de información que se recibe cada segundo por un servidor, se calcula en kbps mediante la siguiente fórmula:

BW=usuarios conectados * Velocidad transferencia kbps

Donde:

BW: Ancho de banda

usuarios conectados: número de usuarios Conectados

Velocidad transferencia kbps: La velocidad de transferencia.

2.5.4. Rendimiento

Permite calcular el número total de solicitudes en bits que se transmiten para su procesamiento en un período de tiempo determinado, su fórmula es:

$$R = \frac{\#bits}{second}$$

Donde:

R: rendimiento

#bits: cantidad de bits procesados

Second: unidad de tiempo

2.5.5. Disponibilidad

Indicador que permite identificar el tiempo de actividad para que los usuarios accedan a los servicios de aplicaciones que ofrece el servidor.

$$Disponibilidad = Uptime / (Uptime + Downtime)$$

Donde:

Uptime: Tiempo de Actividad

Downtime: Tiempo de inactividad

2.6. Criterios éticos.

Los criterios éticos están relacionados al análisis de algoritmos balanceadores de carga en clúster de servidor, por ello se aplicó el siguiente criterio.

a. Criterio de conformidad

El estudio busca estar alineado al reglamento interno de la USS. establece dentro de sus políticas internas respecto a la ética profesional que caracteriza a los profesionales de esta rama de la ciencia, considerando las penalidades sujetas al incumpliendo de objetividad y transparencia.

III. RESULTADOS Y DISCUSIÓN.

3.1. Resultados.

Resultado de Prueba con herramientas Apache Benchmark

Los resultados obtenidos son parte importante para lograr comprobar el rendimiento de un clúster de servidor en la ejecución de pruebas de carga aplicados mediante algoritmos de balanceo de carga.

En un primer escenario de 500 peticiones y 100 usuarios concurrentes, el algoritmo Round Robin (RR) tiene un mejor tiempo de respuesta con un 0.45ms y a su vez consumen menos recursos, todas las técnicas tienen el mismo desempeño en su rendimiento con una tasa de 198kbps, por otro lado en cuanto al consumo de los recursos del CPU y Memoria el algoritmos HC sobre sale con un 11% sobre 13% de las otras técnicas, finalmente la técnica de Weight Roud Robin (WRR) tiene un menor consumo de ancho de banda de 74.53 mbps pero realiza un mayor tiempo de procesamiento de peticiones.

Tabla 4. Rendimiento del servidor escenario 01.

Algoritmo	Tiempo Respuesta	Rendimiento	CPU+M	Ancho Banda
RR	0.45 ms	198 kbps	13%	86.27 mbps
WRR	0.52 ms	198 kbps	13%	74.53 mbps
LC	0.49 ms	198 kbps	13%	79.42 mbps
HC	0.47 ms	198 kbps	11%	81.48 mbps

Nota: Los valores de los indicadores fueron obtenidos en base a los datos obtenidos y registrados en la ficha electrónica de cada una de las pruebas realizadas. Fuente: Elaboración propia.

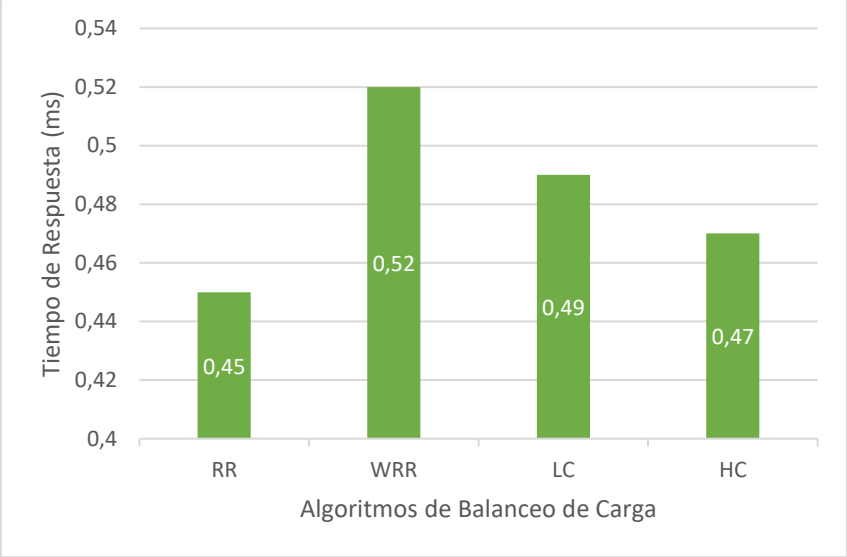


Figura 16. Rendimiento del servidor en base a tiempo de respuesta

Fuente: Elaboración propia.

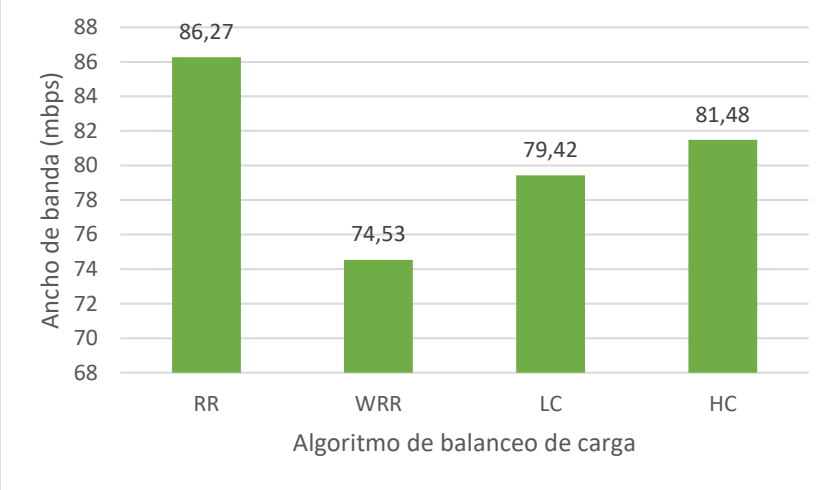


Figura 17. Desempeño del servidor en base a al Ancho de Banda

Fuente: Elaboración propia.

De los datos de la prueba realizada en base los algoritmos RR, WRR, LC y HC de balanceo se muestran una gráfica en cuanto tiempo de duración de la prueba donde se observa la línea de comportamiento en relación al tiempo de respuesta y poder visualizar qué algoritmos tiene una mejor secuencia de atención a las peticiones como se muestra (Figura 18).

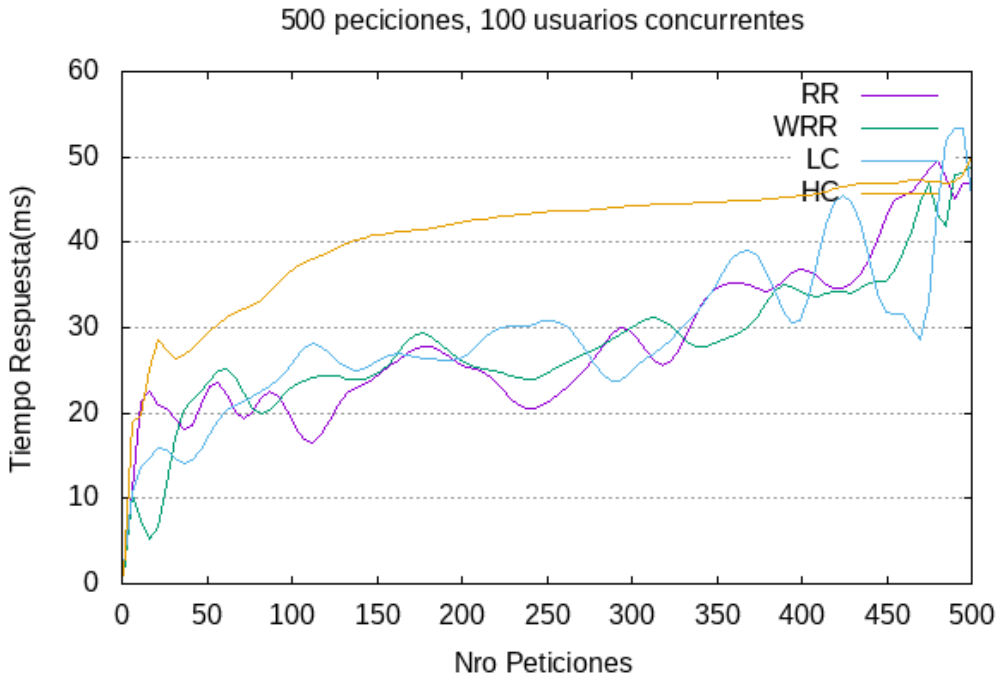


Figura 18. Resultados de tiempo de respuesta de carga en el escenario de 10 usuarios concurrentes y 100 conexiones.

Fuente: Elaboración propia.

Como resultado de indicador del primer escenario

Valor de Indicador: Tiempo de respuesta

Algoritmo	Categoría	Valor
RR	Eficiente	Valor bajo
LC	Deficiente	Valor más alto

Valor de Indicador: Ancho de banda

Algoritmo	Categoría	Valor
WRR	Eficiente	Valor más bajo
HC	Deficiente	Valor más alto

Los resultados del segundo escenario de prueba que se evaluó en base a de 700 peticiones y 300 usuarios concurrentes, donde el RR y HC destaca en cuanto al tiempo de respuesta de 0.48ms mientras que en rendimiento ninguno tiene mejor ventaja con 227.2 kbps, por otro lado en cuanto al consumo de recursos de servidor el HC tiene una menor consumo con 18.1% ante los19.5% de WRR y 20% de RR y 20.5% de LC, en de

igual manera en cuanto al ancho de banda el HC tiene un mejor desempeño porque su consumo es un 241.83 mbps a diferencia de los 241.85mbp de RR, 224.59 mbps de WRR y el LC que tiene un 233.42mbps.

Tabla 5. Rendimiento del servidor escenario 02.

Algoritmo	Tiempo Respuesta	Rendimiento	CPU+M	Ancho Banda
RR	0.48 ms	227.2 kbps	20%	241.85 mbps
WRR	0.52 ms	227.2 kbps	19.5%	224.59 mbps
LC	0.50 ms	227.2 kbps	20.5%	233.42 mbps
HC	0.48 ms	227.2 kbps	18.1%	241.83 mbps

Nota: Los valores de los indicadores fueron obtenidos en base a los datos obtenidos y registrados en la ficha electrónica de cada una de las pruebas realizadas. Fuente: Elaboración propia.

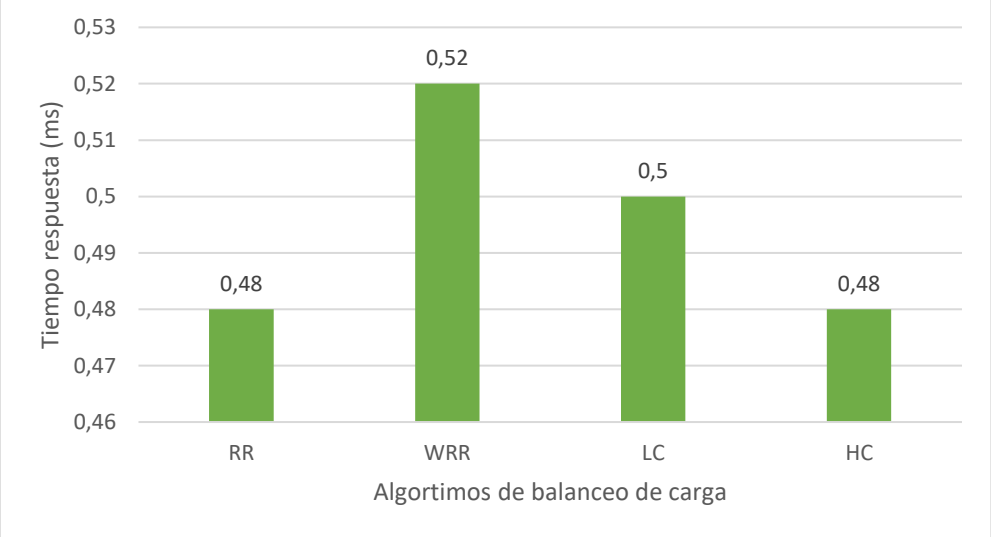


Figura 19. Desempeño del servidor en base al tiempo de respuesta,
Fuente: Elaboración propia.

De los datos de la prueba realizada en base los algoritmos RR, WRR, LC y HC de balanceo se muestran una gráfica en cuanto tiempo de duración de la prueba donde se observa la línea de comportamiento en relación al tiempo de respuesta y poder visualizar qué algoritmos tiene una mejor secuencia de atención a las peticiones como se muestra (Figura 20).

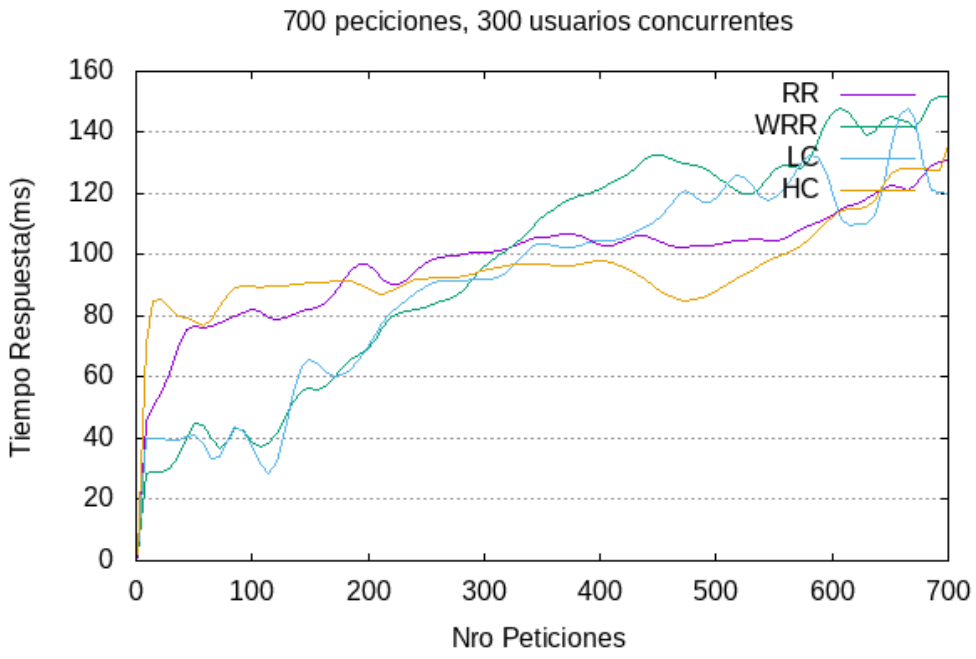


Figura 20. Resultados del tiempo de respuesta de 300 usuarios concurrentes y 700 conexiones.

Fuente: Elaboración propia.

Como resultado de indicador del segundo escenario

Valor de Indicador: Tiempo de respuesta

Algoritmo	**Categoría**	**Valor**
RR, HC	Eficiente	Valor bajo
WRR	Deficiente	Valor más alto

Valor de Indicador: Ancho de Banda

Algoritmo	**Categoría**	**Valor**
HC	Eficiente	Valor más bajo
RR	Deficiente	Valor más alto

Finalmente, en los resultados de la tercera de prueba de balanceo que se evaluó en base a de 1000 peticiones y 400 usuarios concurrentes, en cuanto al tiempo de respuesta el HC tiene un mejor rendimiento con un 0.48ms, en cambio los 4 algoritmos tienen similar desempeño en cuanto al rendimiento con 396kbps, y en relación al consumo de Ancho de banda la técnica de RR y WRR tiene un mejor desempeño con 304.41 mbps ante, los 309.53 mbps y 320.76mbps de LC y HC respectivamente

Tabla 6. Resultados pruebas de balanceo de carga escenario 03.

Algoritmo	Tiempo Respuesta	Rendimiento	CPU+M	Ancho Banda
RR	0.51 ms	396 kbps	30%	304.41 mbps
WRR	0.51 ms	396 kbps	9%	304.41 mbps
LC	0.50 ms	396 kbps	14%	309.53 mbps
HC	0.48 ms	396 kbps	11%	320.76 mbps

Nota: Los valores de los indicadores fueron obtenidos en base a los datos obtenidos y registrados en la ficha electrónica de cada una de las pruebas realizadas. Fuente: Elaboración propia.

De los datos de la prueba realizada en base los algoritmos de balanceo usados se muestran como respondieron durante el tiempo de duración de la prueba (Figura 15).

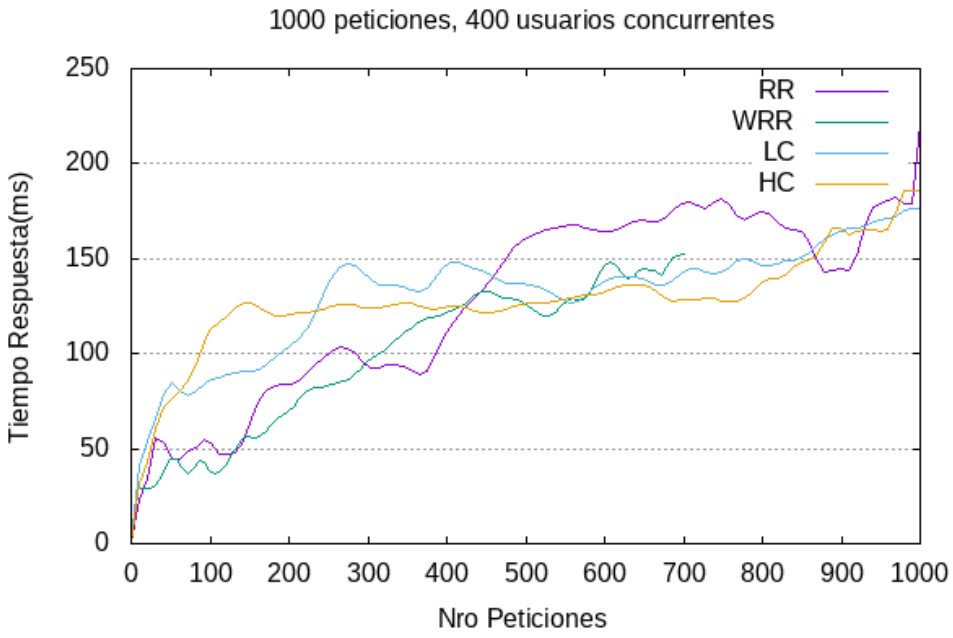


Figura 21. Resultados del tiempo de respuesta de algoritmos balanceo de carga. Fuente: Elaboración propia.

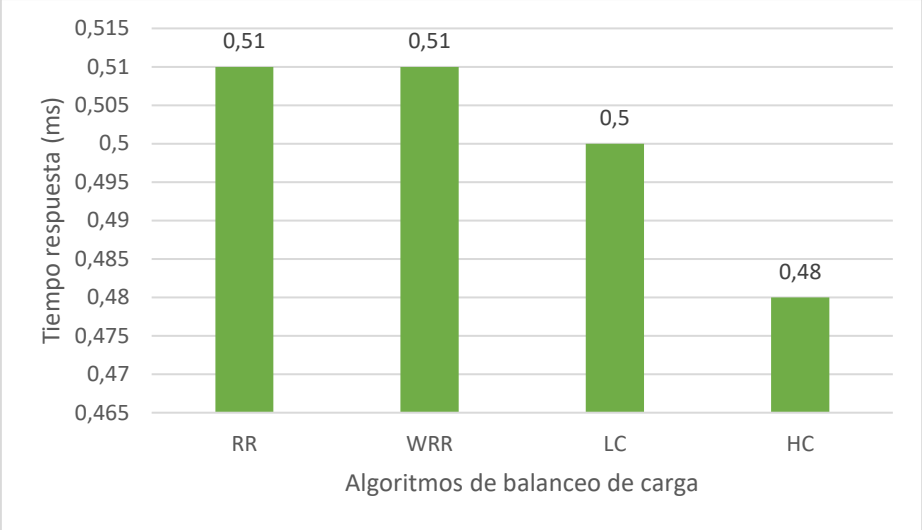


Figura 22. Desempeño del servidor en base a tiempo de rendimiento,

Fuente: Elaboración propia.

Como resultado de indicador del tercer escenario

Valor de Indicador: Tiempo de respuesta

Algoritmo	**Categoría**	**Valor**
HC	Eficiente	Valor bajo
RR,WRR	Deficiente	Valor más alto

Valor de Indicador: Ancho de banda

Algoritmo	**Categoría**	**Valor**
RR,WRR	Eficiente	Valor más alto
HC	Deficiente	Valor bajo

Resultado de Prueba con herramientas Apache Jmeter

Se realizaron pruebas adicionales con esta herramienta para evaluar el rendimiento del servidor con pruebas de mayor carga de peticiones en un intervalo de tiempo definido.

Los resultados de la primera de prueba de balanceo que se evaluó en base a de 500 usuarios con 50 peticiones cada 10 segundos, el RR y HC tienen un mejor desempeño con 8.3 sec, en cuanto a error ninguna petición dios error.

Tabla 7. Rendimiento del servidor escenario 01 con Apache Jmeter.

Algoritmo	Rendimiento	%error	Kbs/sec
RR	8.3 sec	0%	22.62
WRR	7.1 sec	0%	19.42
LC	4.1 sec	0%	11.27
HC	8.3 sec	0%	22.63

Nota: Los valores de los indicadores fueron obtenidos en base a los datos obtenidos y registrados en la ficha electrónica de cada una de las pruebas realizadas. Fuente: Elaboración propia.

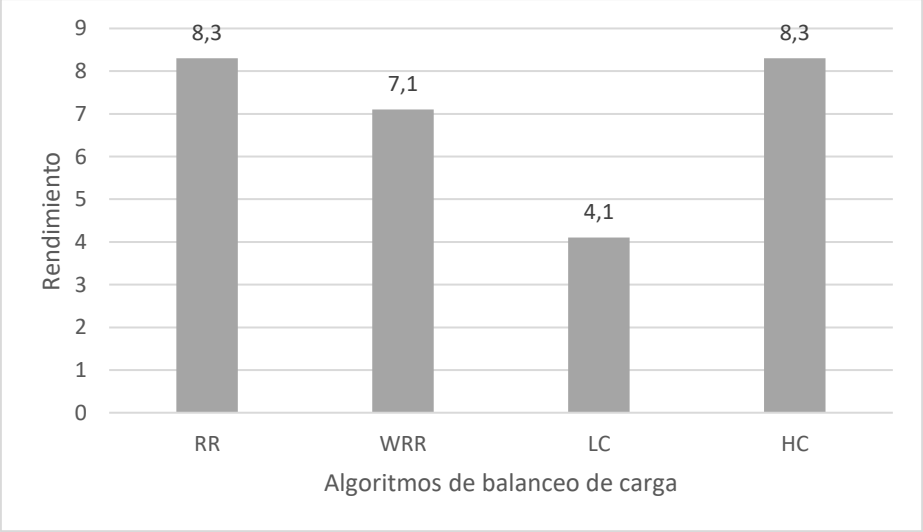


Figura 23. Desempeño del servidor en base a rendimiento,
Fuente: Elaboración propia.

Los resultados de la segunda de prueba de balanceo que se evaluó en base a de 1000 usuarios con 100 peticiones cada 10 segundos, el WRR tiene un mejor desempeño en cuanto al rendimiento con 16.6 sec, en el % de error, el HC presenta mayor error, en cambios WRR y LC tienen 0% de error.

Tabla 8. Rendimiento del servidor escenario 02, con Apache Jmeter.

Algoritmo	Rendimiento	%error	Kbs/sec
RR	15.6 sec	60.40%	39.06
WRR	16.6 sec	0%	45.25
LC	15.0 sec	0%	40.79
HC	15.7 sec	61.70%	39.62

Nota: Los valores de los indicadores fueron obtenidos en base a los datos obtenidos y registrados en la ficha electrónica de cada una de las pruebas realizadas. Fuente: Elaboración propia.

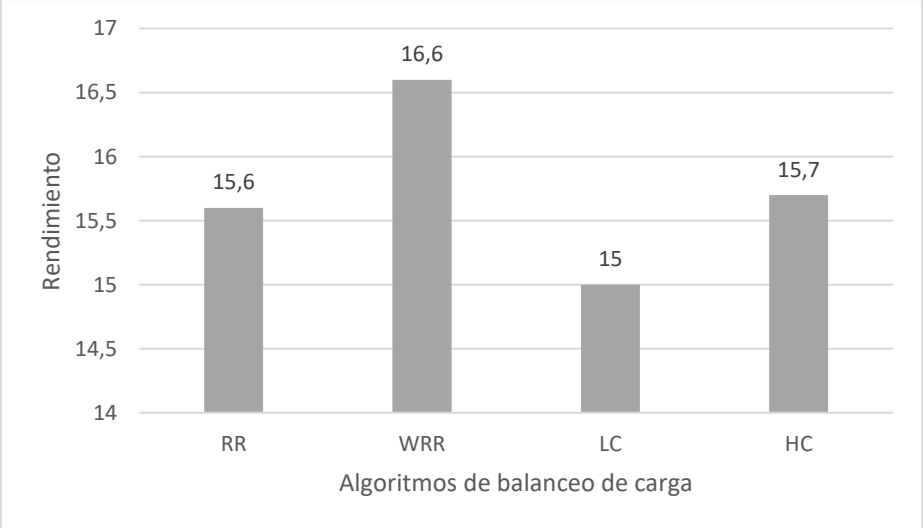


Figura 24. Desempeño del servidor en base a rendimiento,
Fuente: Elaboración propia.

3.2. Discusión.

En referencia al análisis de algoritmos de balanceo de carga implementados en un clúster de servidor, se tiene los siguientes resultados en cuanto al tiempo de respuesta el algoritmo RR fue de 0.51ms, WRR obtuvo un 0.51ms, LC un 0.50ms mientras que HC un 0.48ms, tomando como muestra una carga el escenario de 1000 peticiones con una concurrencia de 400 usuarios, los resultados obtenidos son un aporte debido a que [10] en su investigación evaluó el algoritmo RR y LC en el cual RR obtuvo un tiempo de respuesta de 67.3ms en un escenario de 1000 peticiones con una concurrencia de 100 usuarios. Si bien el tiempo de respuesta de nuestra prueba es más eficiente es importante precisar que nuestro entorno de aplicación del clúster de servidor fue en ambiente máquinas virtual a comparación [10] que fue aplicado en minicomputadoras Raspberry Pi y las herramientas de balanceo de carga utilizados en nuestra prueba es Nginx en comparación que utilizó Haproxy, Teniendo en cuenta estas condiciones también se obtuvo un mejor desempeño en cuanto al indicador de rendimiento de servidor con una tasa rendimiento de 396 kbps.

Del mismo modo estos resultados son importantes porque [9], evaluó estos dos indicadores de tiempo de respuesta y rendimiento obteniendo resultados en base a un escenario de las pruebas de 1000 peticiones, donde el tiempo de respuesta del algoritmo WRR obtuvo un tiempo de respuesta de 1.16ms y el rendimiento el WRR obtuvo 446.199kbps, a comparación de nuestro resultado en el tiempo de respuesta del algoritmos HC fue de 048ms y en el rendimiento obtuvo un 396kbps, si bien ambas pruebas se ejecutaron en escenarios diferentes los resultados nos dan proporciona de que se conoce que el algoritmo WRR tiene un mejor desempeño.

En referencia al análisis de algoritmos de balanceo de en cuanto al consumo de ancho de banda se tiene los siguientes resultados el algoritmo RR es de 304.41 mbps, WRR obtuvo un 304.41mbps, LC un 309.53mbps mientras que HC un 320.76mbps tomando como muestra una carga el escenario de 1000 peticiones con una concurrencia de 400 usuarios, los resultados obtenidos son un aporte debido [14] en su estudio evaluó técnicas dinámicas como el RR y WRR en un entorno de prueba de 100 conexiones con intervalos de tiempo de 1 segundo en el que RR y WRR tuvieron un promedio de ancho de banda de 210kbps, si bien nuestro resultados tienen un mayor consumo de ancho de banda, hay diferencias entre el tipo de herramientas y utilizadas para implementar el balanceador de carga, por su parte se basó en un entorno de red definido por software y otras aplicaciones como Mininet o controladores POX y OpenFlow. Por nuestra parte se implementó un balanceador de carga en un clúster de servidores web.

Sobre el análisis de consumo de recursos como es de CPU y memoria los resultados obtenidos son los siguientes RR un 30%, WRR 9%, LC 14% y HC 11% tomando como muestra una carga el escenario de 1000 peticiones con una concurrencia de 400 usuarios. Si bien estos resultados son de importancia para medir el rendimiento de servidor y validar su capacidad de procesamiento no se puede comparar a otras investigaciones por lo que no lo han considerado dentro de su estudio como parte del rendimiento del servidor de balanceo.

3.3. Aporte de la investigación.

El método propuesto se realizará mediante etapas las cuales nos permitirán validar que algoritmos de balanceo de carga permitan mejorar la disponibilidad de un servidor, para lo cual se realizará una selección de algoritmos en base a la revisión sistemática de artículos la cual permite definir nuestra población de algoritmos y según criterios de comparación que tienen relación a nuestros indicadores de VI y VD que permitieron obtener nuestra muestra de investigación, del mismo modo identificando las diversas topologías de arquitectura de clúster de servidor, lo cual nos permitirá definir un esquema de clúster y validar su funcionamiento implementado las técnicas de balanceo de carga seleccionados en nuestra muestra.

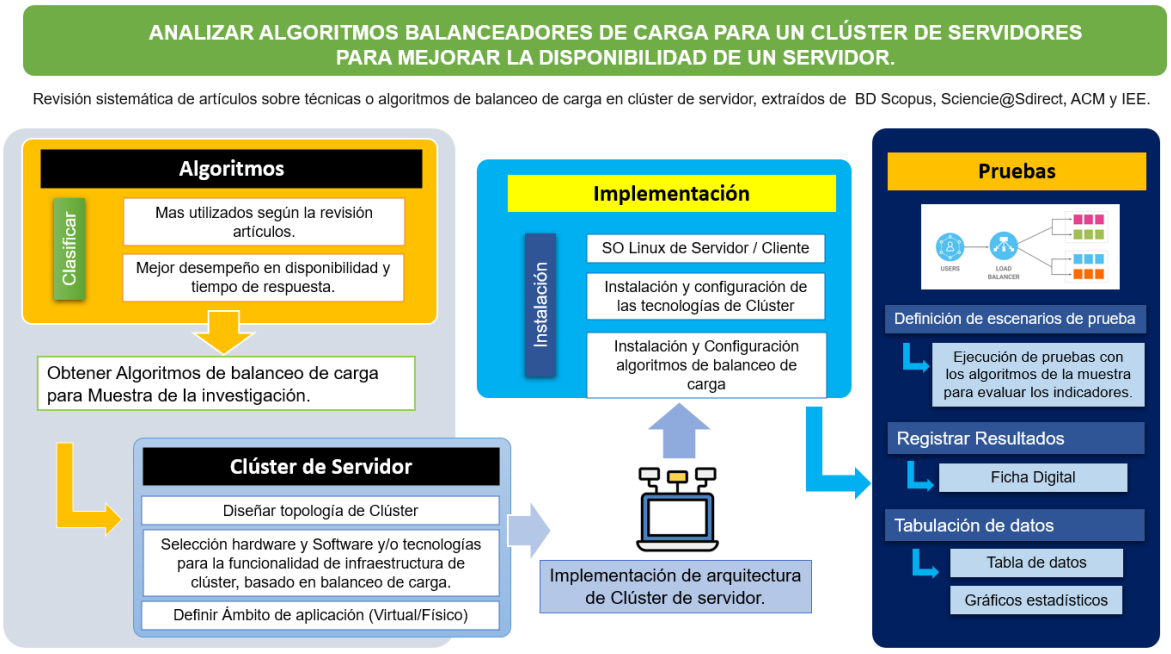


Figura 25. Esquema de implementación de objetivos específicos.
Fuente: Elaboración propia.

A continuación, detallamos el procedimiento realizado en cada una de los objetivos específicos.

Obj 1.- Seleccionar algoritmos balanceadores de carga más utilizados y de mejor desempeño.

Para llevar a cabo este objetivo, primero realizó una revisión sistemática de la literatura, proceso que consistió en la búsqueda de algoritmos de balanceo de carga el cual se pasó por diferentes etapas, desde la panificación de preguntas sobre los indicadores o parámetros para medir el rendimiento, tiempo de respuesta y otros valores de respuesta del servidor, así como los escenarios de aplicación, el hardware y software y los lenguajes de programación utilizados.

Este estudio del mapeo sistemático, de la búsqueda de los artículos se realizó en bases de datos científicas mediante una búsqueda de cadena elaborada en base a palabras clave del tema en investigación, En la búsqueda se hicieron uso de cuatros bases de datos científicas, ACM Digital Library, IEEE Digital Library, Science@Direct y Scopus, con el propósito de obtener artículos revisados y publicados en revistas, en simposios de renombre y conferencias.

Luego se procedió a la evaluación de cada artículo a fin de determinar su importancia y si estos contribuyen a la investigación de los cuales se obtuvieron los datos más importantes.

Tabla 9. Información de datos extraídos de los artículos

Nro	Datos	Descripción.
1	Año	Año de publicación del artículo
2	Título	Título del trabajo
3	Autores	Los autores del artículo
4	País	País de afiliación de los autores
5	Canal de publicación	El canal a través del cual se publica el artículo.

6	Tipo de Publicación	Revista / Conferencia / Taller / etc.
7	Fuente de Publicación	Academia / Industria / Ambos
8	Tipo de Artículo	Tipo basado en el esquema de clasificación
9	Contribuciones en artículo	Principales contribuciones del trabajo
10	Resumen	Resumen del artículo

El resultado del protocolo de búsqueda, se pudo recuperar un total de 322 artículos, luego se clasificaron en base a los títulos de los artículos que estaban relacionados a la investigación quedando 116 artículos para la siguiente etapa de revisión, luego se descartaron los duplicado quedando 92 artículos de los cuales se dieron lectura a los resúmenes, resultados y conclusiones, de los cuales se procedió a dar lectura completa resultando con 79 artículos aptos para realizar la evaluación de calidad en base a los procedimientos, técnicas tecnologías que en estas investigaciones hicieron uso. Pasado el proceso de calidad se seleccionaron 59 artículos para su inclusión en el estudio.

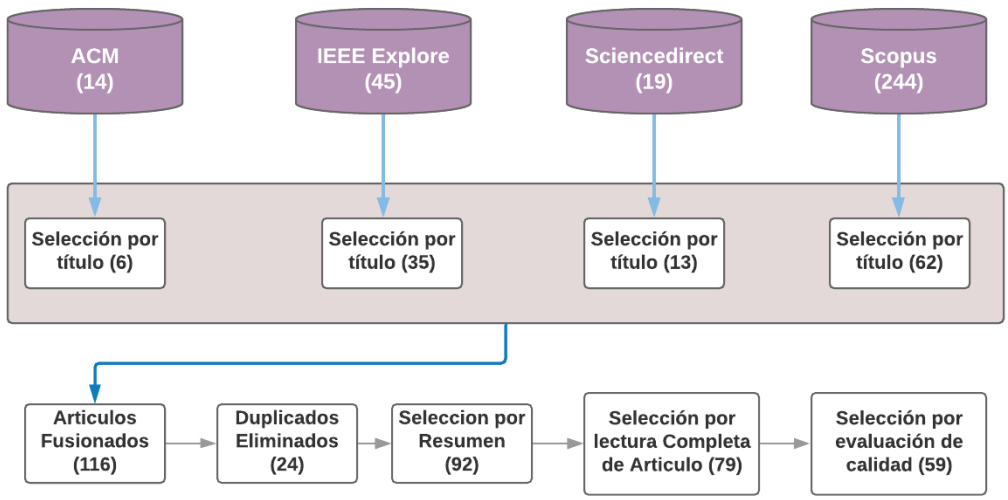


Figura 26. Proceso de selección de las investigaciones relevantes.

Tabla 10: Lista de artículos seleccionados

#	Autor	Año	Publicación	Ámbitos de uso
1	Giri, Nupur and Kukreja, Vikas and Panchi, Dinesh and Sajnani, Jatin and Seedani, Hitesh	2018	Conference	Servidores en SDN
2	Prasetijo, Agung B. and Widianto, Eko D. and Hidayatullah, Ersya T.	2017	Conference	Servidores Web
3	Kumar, Ankit and Kalra, Mala	2016	Conference	Centro de Datos en la Nube
4	Rai, Himanshu and Ojha, Sanjeev Kumar and Nazarov, Alexey	2021	Conference	Centro de Datos en la Nube
5	Dhahbi, S. and Berrima, M. and Al-Yarimi, F.A.M.	2021	Journal	Centro de Datos en la Nube
6	Suherman, S. and Aziz, M. and Nababan, E.B.	2019	Journal	Servidor de Video
7	Kumar, P.J. and Sivakumar, N. and Prabhu, J. and Ramesh, P.S. and Suganya, P.	2019	Journal	Centro de Datos en la Nube
8	Cao, Y.	2021	Conference	Servidores Web
9	Zatwarnicki, K.	2021	Journal	Servidores Web
10	Chiang, M.-L. and Cheng, H.-S. and Liu, H.-Y. and Chiang, C.-Y.	2020	Journal	Servidores en SDN
11	Noman, H.M. and Jasim, M.N.	2021	Conference	Servidores en SDN
12	Rukmini Bhat, B. and Sneha, N.S. and Bhat, K. and Kamath, C.C. and Naik, C.	2021	Conference	Servidores en SDN
13	Jiang, X. and Yang, H. and Yang, Y. and Chen, Z.	2021	Journal	Servidores Web
14	Yu, A. and Yang, S.	2020	Journal	Servidores Web
15	El Karadawy, A.I. and Mawgoud, A.A. and Rady, H.M.	2020	Conference	Centro de Datos en la Nube
16	Qin, E. and Wang, Y. and Yuan, L. and Zhong, Y.	2020	Conference	Servidores Web
17	Farhoudi, M. and Habibi, P. and Sabaei, M.	2019	Simposio	Servidores en SDN
18	Liu, G. and Wang, X.	2019	Conference	Servidores Web
19	Wilson Prakash, S. and Deepalakshmi, P.	2018	Journal	Servidores Web
20	Chakravarthy, V.D. and Amutha, B.	2019	Journal	Servidores en SDN
21	Wen, Z. and Li, G. and Yang, G.	2018	Conference	Servidores Web
22	Suwandika, I.P.A. and Nugroho, M.A. and Abdurahman, M.	2018	Conference	Servidores en SDN

23	Li, J. and Nie, Y. and Zhou, S.	2018	Conference	Servidores Web
24	Rishabh, K. and Angadi, K. and Chegu, K. and Harikrishna, D.S. and Ramya, S.	2018	Conference	Servidores en SDN
25	Li, R. and Li, Y. and Li, W.	2018	Conference	Servidores Web
26	Ramana, K. and Ponnavaikko, M.	2018	Journal	Servidores Web
27	Tennakoon, D. and Chowdhury, M. and Luan, T.H.	2018	Journal	Servidores en SDN
28	Alam, F. and Thayananthan, V. and Katib, I.	2016	Conference	Servidores Web
29	Saifullah, M.A. and Mohamed, M.A.M.	2016	Conference	Servidores en SDN
30	Aditya, B. and Juhana, T.	2016	Conference	Servidores BD
31	Nair, A. and Binya Mol, M.G. and Nair, N.S.	2016	Journal	Servidores en SDN
32	Xu, Z. and Wang, X.	2015	Conference	Servidores Web
33	Chen, W. and Shang, Z. and Tian, X. and Li, H.	2015	Journal	Servidores en SDN
34	Xiong, Z. and Liu, W.	2015	Journal	Servidores Web
35	Zhao, H. and Zhou, C.-L. and Jin, B.-Z.	2015	Journal	Servidor de Video
36	Zhu, Liangshuai and Cui, Jianming and Xiong, Gaofeng	2019	Conference	Cluster de Servidor
37	Yi, Chang and Zhang, Xiuguo and Cao, Wei	2018	Conference	Servidor Microservicios
38	Alazzam, Hadeeel and Alsmady, Abdulsalam and Mardini, Wail and Enizat, Amira	2019	Conference	Centro de Datos en la Nube
39	Chen, Zhidong and Zhang, Hailong and Yan, Jinyao and Zhang, Yuan	2018	Conference	Centro de Datos en la Nube
40	Maniza Hijab and Avula Damodaram	2020	Conference	Cluster de Servidor
41	Handur, Vidya S. and Belkar, Supriya and Deshpande, Santosh and Marakumbi, Prakash R.	2019	Conference	Centro de Datos en la Nube
42	Swarnakar, Soumen and Kumar, Ritik and Krishn, Saurabh and Banerjee, Chandan	2020	Conference	Centro de Datos en la Nube
43	Mesbahi, Mohammad Reza and Hashemi, Mahnaz and Rahmani, Amir Masoud	2016	Conference	Centro de Datos en la Nube
44	Sidana, Shubham and Tiwari, Neha and Gupta, Anurag and Kushwaha, Inall Singh	2017	Conference	Centro de Datos en la Nube
45	Rai, Himanshu and Ojha, Sanjeev Kumar and Nazarov, Alexey	2021	Conference	Centro de Datos en la Nube

46	Vidya, S. Handur and Prakash, R. Marakumbi	2020	Conference	Centro de Datos en la Nube
47	Kaushik Mishra and Jharashree Pati and Santosh Kumar Majhi	2020	Journal	Centro de Datos en la Nube
48	Farias, J.V.O. and Coutinho, E.F. and Bezerra, C.I.M.	2021	Conference	Servidores en SDN
49	Velde, V. and Rama, B.	2018	Conference	Centro de Datos en la Nube
50	Surbhi and Oshin and Bhatt, Mahesh Chandra	2018	Conference	Cluster de Servidor
51	Murti, Krisna Wahyu and Riza, Tengku Ahmad and Mulyana, Asep	2020	Conference	Servidores Web
52	Prakash, S. Wilson and Deepalakshmi, P.	2017	Conference	Cluster de Servidor
53	Li, Jingmei and Yang, Linfeng and Wang, Jiaxiang and Yang, Shuang	2019	Conference	Centro de Datos en la Nube
54	Jadhav, Kiran A and Mulla, Mohammed Moin and Narayan, D. G	2020	Conference	Servidores en SDN
55	Liu, Hsien-Yi and Chiang, Ching-Yi and Cheng, Hui-Sheng and Chiang, Mei-Ling	2018	Conference	Servidores en SDN
56	Saraswati, I. and Praptodiyono, S. and Pramudyo, A.S. and Kurniawan	2019	Conference	Servidores Web
57	Afrianto, Y. and Sukoco, H. and Wahjuni, S.	2018	Journal	Servidores Web
58	Rohadi, E. and Amalia, A. and Prasetyo, A. and Rahmat, M.F. and Setiawan, A. and Siradjuddin, I.	2020	Conference	Cluster de Servidor
59	Hamdani, M. and Aklouf, Y. and Chaalal, H.	2020	Conference	Servidores en SDN

Fuente: Elaboración Propia.

Todas estas etapas ejecutadas están direccionadas en contribuir al cumplimiento del objetivo de obtener algoritmos balanceadores de carga más utilizados y de mejor desempeño, de los cuales para identificarlos a cada uno se clasificaron en un listado con su abreviatura.

Tabla 11. Algoritmos balanceadores de carga y su identificador.

Algoritmo	**Abreviatura**
Round Robin	RR
Random	R
Least Conecction	LC

Least-Loading	LL
Weighted Round Robin	WRR
Throttled	THR
Weighted ramdon selection	WRS
Weighted Least Connection	WLC
IP Hash	IPH
Load Balancer Server	LB
Hash Consistente	HC
Dynamic Weighted Least Connection	DWLC
Dynamic Round Robin	DRR
First Come First Served	FCFS
Software Defined Network	SDN
Dynamic Load Balancer Server	DSLB

Fuente: Elaboración Propia

Tabla 12. Algoritmos de balanceo de carga más utilizados según la revisión sistemática de la literatura.

Algoritmo	**Referencia**	**N° Citas**
RR	Noman & Jasim [14], R. Li et al [48], Jezreel Ian C. Manuel et al [62], G. Liu & Wang [63], Chiang et al [13],W. Chen et al.[64], Prakash & Deepalakshmi, [65], Jadhav et al. [12], H. Y. Liu et al. [11], A. Kumar & Kalra [66],Rai et al. [67], Suherman et al. [22], P. J. Kumar et al. [68], Zatwarnicki, [69], El Karadawy et al. [70], Mesbahi et al. [71], Velde & Rama. [72], Hamdani et al.[19], J. Li et al. [68], Suwandika et al.[73], Wilson Prakash & Deepalakshmi, [21], Farhoudi et al. [74], Rohadi et al. [10], Saraswati et al. [9], Prasetijo et al.[75]	25
R	Noman & Jasim [14], Chiang et al. [13], Yu & Yang, [76], W. Chen et al. [64], Prakash & Deepalakshmi, [65], Jadhav et al. [12], Suherman et al. [22], J. Li et al.[68], Wilson Prakash & Deepalakshmi. [21]	9
LC	Noman & Jasim,[14], R. Li et al. [67], Xiang et al.[77], G. Liu & Wang [63], Yu & Yang. [76], Zhu et al.[78], Suwandika et al.[73]), Rohadi et al.[10], Prasetijo et al.[75]	9

WRR	Noman & Jasim. [14], Suherman et al. [22], Z. Chen et al.[79], Aditya & Juhana. [80], Ramana & Ponnavaikko. [81], Jiang et al.[82], Saraswati et al.[9]	7
LL	H. Y. Liu et al.[11], Zatwarnicki. [69], El Karadawy et al.[70]	3
THR	A. Kumar & Kalra.[66], Rai et al.[83], Mesbahi et al.[71], Hamdani et al.[19]	3
WRS	Chiang et al.[13], H. Y. Liu et al.[11]	2
WLC	Xiang et al. [84], Jiang et al.[85]	2
IPH	Suherman et al. [22]), Suwandika et al.[73]	2
SBL	W. Chen et al. [13]	1
LB	Noman & Jasim.[14]	1
DWLC	R. Li et al. [67]	1
DRR	Jezreel Ian C. Manuel et al. [62]	1
FCFS	Jezreel Ian C. Manuel et al. [62]	1
SDN	Farhoudi et al. [74]	1
DSLB	Prakash & Deepalakshmi. [65], J. Li et al.[68]), Wilson Prakash & Deepalakshmi.[65]	3

Fuente: Elaboración Propia

Se clasificaron los artículos que en su metodología implementaron o evaluaron estos dos criterios, lo cual se describe para cada artículo que cumple esta condición y su escenario de prueba físico o virtual.

Según estas características se identificaron 09 artículos de los cuales se describen en las siguientes tablas:

Tabla 13. Los indicadores y los resultados del rendimiento y tiempo respuesta.

Referencia	[75]	
Escenario prueba:	400 peticiones /s (mayor escenario)	
Ámbito / Tecnologías	Físico / Haproxy	
Criterios	Estrategias	
	RR	LC
Tiempo de respuesta (ms)	< 1000 ms	>1000 kb/s
Rendimiento (kb/s)	< 500 kb/s	>500 kb/s
Conexiones fallidas (cf)	>100	<90

Referencia	[9]	
Escenario prueba:	5000 peticiones /s (mayor numero)	
Ámbito / Tecnologías	Virtual / ipvsadm	
Criterios	Estrategia	
	WRR	RR
Tiempo de respuesta (ms)	28.75 s	47.79 s
Rendimiento (kb/s)	923.95 kb/s	906.46 kb/s
Solicitud de error	1.03 %	1.10%

Referencia	[10]	
Escenario prueba:	500 usuarios / 1000 Peticiones /s (mayor numero)	
Ámbito / Tecnologías	Físico/mini Raspberry Pi / HAProxy	
Criterios	Estrategia	
	RR	LC
Tiempo de respuesta (ms)	182.3 ms	350ms
Rendimiento (kb/s)	79.62 kbps	55.40 kbps
Solicitud de error	0%	0%

Referencia	[74]	
Escenario prueba:	1000 (8-pod fat-tree)	
Ámbito / Tecnologías	Virtual / Mininet /Floodlight,	
Criterios	Estrategia	
	RR	SDN
Tiempo de respuesta (ms)	>1200	<400 ms
Rendimiento (kb/s)	<6 mb/s	>7mb/s
Solicitud de error	-	-

Referencia	[21]		
Escenario prueba:	Concurrencia de 1000 usuarios / 7000 N° de Solicitudes por Segundo		
Ámbito / Tecnologías	Virtual / Mininet / OpenFlow.		
Criterios		Estrategia	
	R	RR	DS-LB
Tiempo de respuesta (ms)	>30	30	<30
Rendimiento (mbps)	<1.5 mbps	1.5mbps	>2mbps
Solicitud de error (Sol/sec)	3500	>3000	<3000

Referencia	[73]		
Escenario prueba:	Solicitudes aleatorias (20 clientes)		
Ámbito / Tecnologías	Físico/Iperf		
Criterios		Estrategia	
	H	LC	RR
Tiempo de respuesta (ms)	50-65	60-100	54-64
Rendimiento (kb/s)	1.23mbps	1.11mbps	1.17mbps
Solicitud de error	-	-	-

Referencia	[85]		
Escenario prueba:	N° peticiones 1500 / Numero de nodos (8)		
Ámbito / Tecnologías	Virtual / Jmeter / Linux		
Criterios	Estrategia		
	WRR	WLC	HC
Tiempo de respuesta (ms)	>3000	>3000	<3000
Rendimiento (kb/s)	<3000	<3000	>3000
Solicitud de error	0-25%	0-25%	0-25%

Referencia	[81]		
Escenario prueba:	1000 solicitudes		
Ámbito / Tecnologías	Físico / Windows		
Criterios	Estrategia		
	WRR	CAP	MCLB
Tiempo de respuesta (ms)	>9000	>7000	3000
Rendimiento (kb/s)	<200	<800	>1200
Solicitud de error	<30	<20	<10

Referencia	[80]	
Escenario prueba:	700 sub procesos	
Ámbito / Tecnologías	Virtual / Linux	
Criterios	Estrategia	
	WRR	WRROPT
Tiempo de respuesta (ms)	Sin Conexión	>6000
Rendimiento (kb/s)	Sin conexión	100
Solicitud de error	Error	-

A continuación, se elaboró una Tabla resumen de resultados de comparación de tiempo de respuesta y rendimiento de los algoritmos con mejor desempeño.

Tabla 14. Resumen de algoritmos y sus valores de rendimiento y tiempo de respuesta.

Algoritmo o Técnica de balanceo de carga	**Numero Peticiones**	**Criterios desempeño**		
		Tiempo de respuesta	Rendimiento	Conexiones fallidas
WRR	5000	28.75ms	923.95kbs	1.03%
CH	1500	<3000	>3000	0-25%
RR	1000	182.3ms	79.62kbps	0%
SDN	1000	<400 ms	>7mbs	
MCLB	1000	3000ms	>1200	<10
R	1000	13.8ms	92.41kbs	5.3
WRROP	700	>6000	100	
IPH	400	50-65 ms	1.23 mbps	
LC	400	>1000ms	>500 kb/s	<90

Fuente: Elaboración propia.

Una vez identificado la población de algoritmos y cómo estos se comportaron en relación a los ambientes de pruebas, se seleccionaron los algoritmos más usados y que obtuvieron mejor resultado en criterios que mejoran la disponibilidad de los servicios de un servidor y el tiempo de respuesta, mediante este procedimiento se obtiene nuestra muestra para realizar su implementación y pruebas posteriores.

Lista de algoritmos, los cuales se utilizaron para las pruebas de análisis de balanceo

Weighted Round Robin (WRR)
Consistent Hashing (CH)
Round Robin (RR)
Least Conection (LC)

Obj 2.- Determinar arquitectura de Clúster de servidores para realizar las comparaciones de los algoritmos balanceadores de carga.

Para poder definir el esquema o arquitectura de clúster a implementar, se referencio a los ambientes de prueba que fueron empleados en las diferentes metodologías usadas para las pruebas en los diferentes artículos revisados y agregando otros elementos que sumarán a mejorar el funcionamiento del clúster de servidor según las investigaciones realizadas, para definir la arquitectura se definieron en etapas las cuales son:

- Diseñamos la arquitectura o topología del clúster de servidor.

El diseño considera un Back-end, al cual se conectan mediante un Switch al Front-end en los cuales están implementados las aplicaciones web, de los cuales sería el medio de acceso de atención a las peticiones que los usuarios enviarán desde el internet.

El Frontend consta de 3 host para servidores de aplicaciones de los cuales 1 hace de backup para ser usado cuando el otro nodo no esté disponible, en los cuales se ha

instalado un sistema de gestión de proyectos, y para el Backend de un cuarto servidor (nodo 4), en el cual estará implementada la BD del sistema, en la que el funcionamiento del balanceo de carga se aplicaría en el Front-end, con el fin de distribuir las peticiones hacia el back-end.

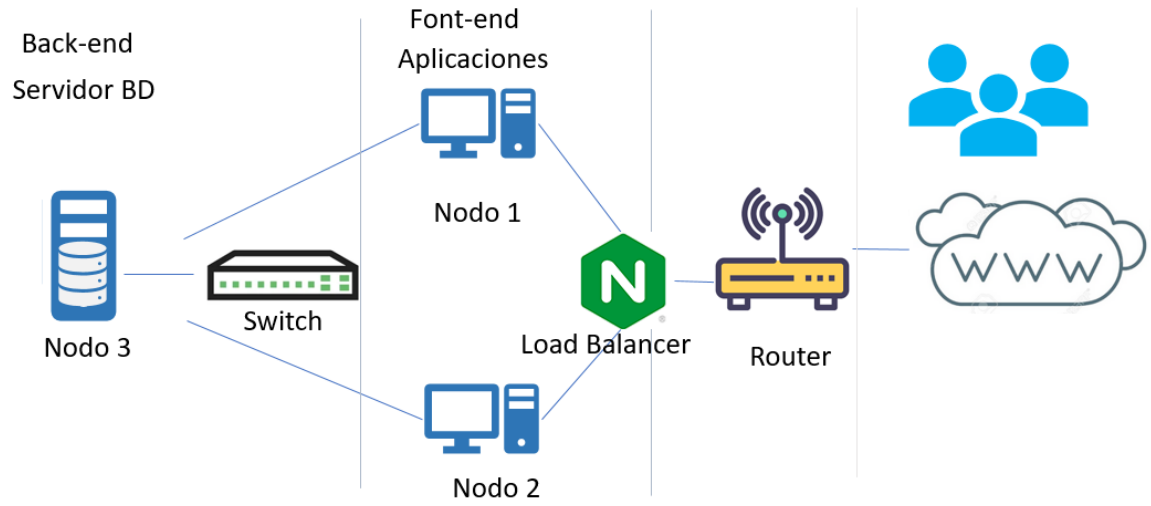


Figura 27: Diseño de arquitectura de clúster de servidor.
Fuente: Elaboración propia, basado en [9], [10]

El Nodo 1 y Nodo 2 tiene como fin configurarlos para atender las peticiones de manera vertical y de esta manera obtener una alta disponibilidad entre ellos. para el acceso de los usuarios externos.

Se diseñó esta arquitectura con el fin de separar las aplicaciones del BD, debido a que al separar estos componentes permite disminuir que las solicitudes de acceso a la aplicación se concentren en un solo Nodo, y de esta manera se aumentará la disponibilidad y mejorar el rendimiento, para mejorar la funcionalidad del Clúster de Servidor facilitará la tolerancia a posibles fallos, asimismo permitirá aumentar la seguridad de resguardo a la información que se va almacenando.

Tabla 15. Características de los componentes de los nodos del clúster de servidor.

Componente	Uso
Servidor de Balanceo de carga	Para direcciones las peticiones de los clientes
2 nodos de Aplicaciones web 1 nodo de Backup	Frontend del Cluster
1 nodo de Servidor de BD	Backend del Cluster
Virtual Box	Virtualización de Máquinas Virtuales
Sistema Operativo (Linux)	Plataforma a desplegar en los Nodos del clúster
Nginx	Servidor Web
Software Clúster de servidor	Complementar la arquitectura de clúster.
Software de balanceo de carga (Algoritmos)	Algoritmos seleccionados
Software de pruebas de estrés	Aplicaciones para comparar el desempeño del clúster en respuesta a las pruebas de los algoritmos de balanceo de carga

Asimismo permitirá, aumentar el número de usuarios conectado, el número de solicitudes para aumentar el tráfico de la red y disminuir costos, [77]

- Definimos los componentes de hardware y software necesarios para el balanceo de carga.

Para completar el funcionamiento de la arquitectura es indispensable hacer uso de tecnologías adicionales para lograr el funcionamiento de un clúster de servidor, se hará de tecnologías.

Componentes Hardware y Software

- Característica de los nodos

Nodo Servidor Balanceo de Carga

SO: Linux Ubuntu 20.04 (64 bits)

RAM: 3GB

Disco Duro: 25GB

Interfaz de red (Conectado a la red Interna)

Nodo2 Aplicaciones (Server 1 Server 2)

SO: Linux Ubuntu 20.04 (64 bits)

RAM: 2GB

Disco Duro: 10GB

Interfaz de red (Conectado a la red Interna)

Nodo2 Servidor BD

SO: Linux Ubuntu 20.04 (64 bits)

RAM: 2GB

Disco Duro: 15GB

Interfaz de red (Conectado a la red Interna)

- Implementamos la topología de clúster de servidor.

Para este proceso, de realizar la selección de herramientas que permitió desplegar la arquitectura de clúster diseñada, la aplicación usada es Oracle VM VirtualBox, en el cual se consideró lo siguiente:

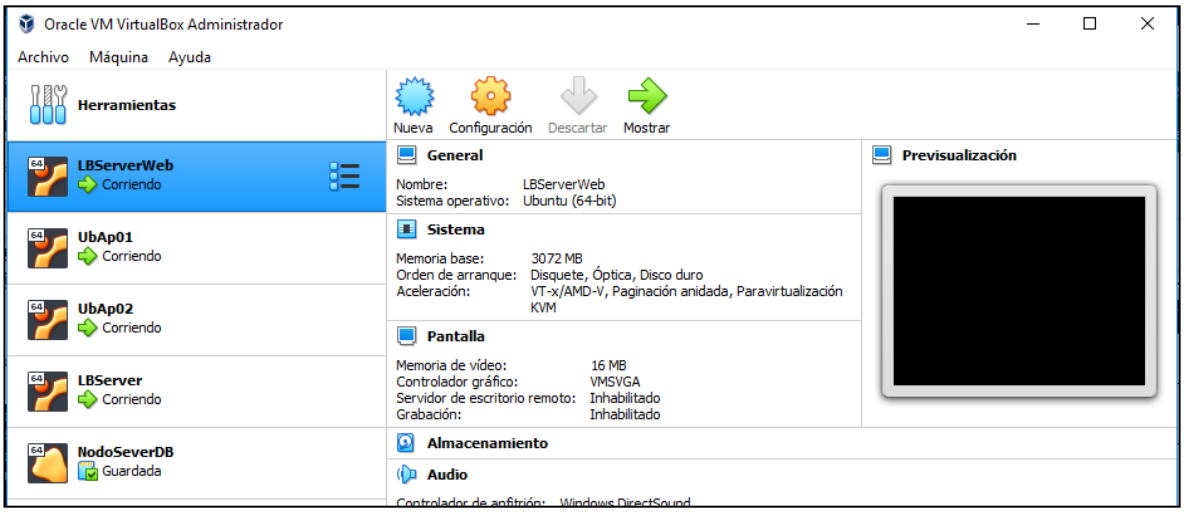


Figura 28: Software Oracle VM VitualBox, creación de esquem de clúster de servidor

Fuente: Elaboración Propia.

Instalación de Servidor de Balanceo de carga

Para nuestro balanceador de carga se instaló y configuro el sistema operativo Linux en su distribución ubuntu-20.04.2.

Una vez instalado se procedió a realizar la instalación del servidor de balanceo Nginx para lo cual se ejecutó la actualización del sistema, para luego realizar instalación del paquete para que se instale con todas sus dependencias creando un servicio *nginx.service* que debe ser configurado para que habilitado e inicie automáticamente.

```
~$ sudo apt update
~$ sudo apt install – y nginx
~$ sudo systemctl start nginx
```

Se configura el firewall añadiendo unas reglas para el acceso desde la red, para las peticiones http y https.

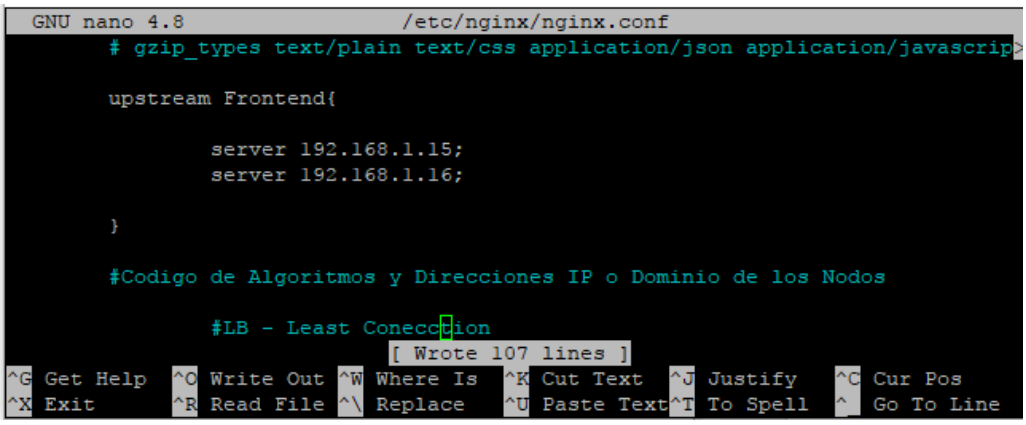
```
~$ sudo ufw allow http
~$ sudo ufw allow https
```

Asimismo, se habilitoó el acceso SSH y el puerto para acceder mediante un cliente SSH y realizar las configuraciones del balanceo de carga.

Accedemos el servidor mediante un cliente SSH, y mediante el usuario root, ingresamos a configurar el archivo *nginx.conf*.

```
~$ sudo nano /etc/nginx/nginx.conf
```

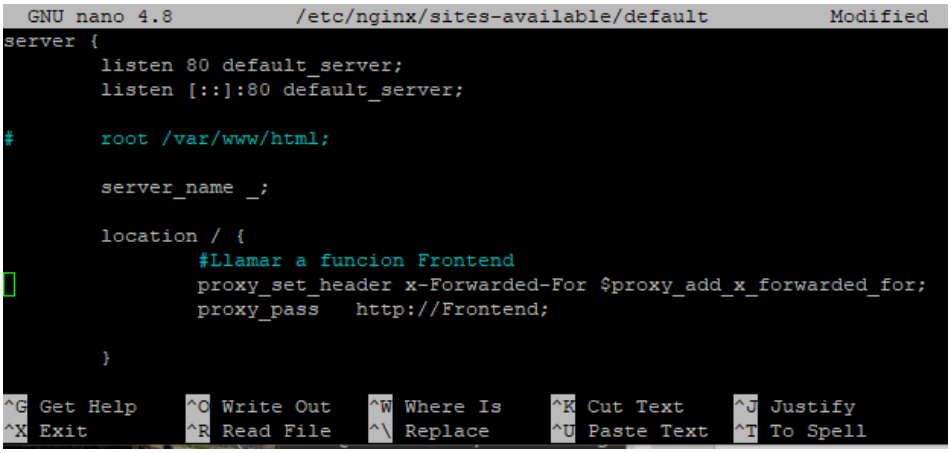
Nos desplazamos a la opción de HTTP y creamos la función *Fronted con upstream*, en la cual indicaremos los números de direcciones IP o dominios de nuestros nodos o servidores web que forman parte de Frontend y guardamos los cambios.



Asimismo, configuramos el direccionamiento por defecto a la función Frontend, para direccionar las peticiones que ingresan por parte de los usuarios.

```
~$ sudo nano /etc/nginx/sites-available/default
```

En este archivo, configuración en location la llamada a la funcion creada en el archivo de *nginx.conf*



Para tener listo la configuración de balanceo de carga se realiza un test para validar si la configuración es correcta y luego se reinicia el servicio para que quede activo.

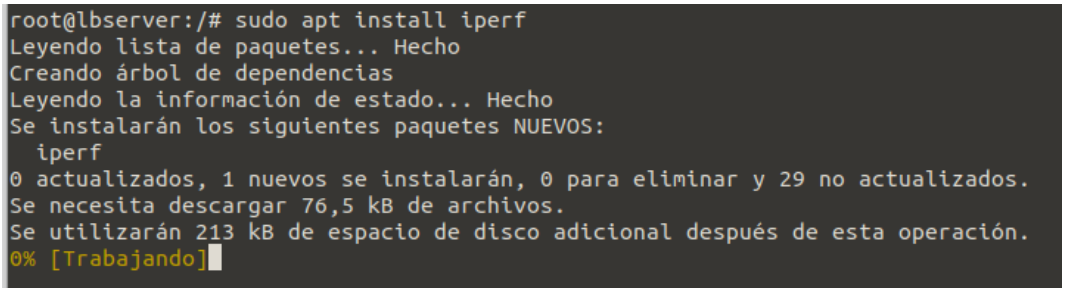
```
~$ sudo nginx -t
~$ sudo systemctl restart nginx
```

Al ejecutar estos comandos ya se tiene nuestro servidor balanceador de carga activado y podemos comprobar accediendo desde un navegador la dirección IP del Servidor, que saldría un error 502 el cual hace referencia que un no está completo nuestro balanceador, porque no está direccionan al no encontrar las direcciones configuradas, debido a que aún nos falta implementar los servidores al cual hace referencia la función *upstream Frontend{}.*

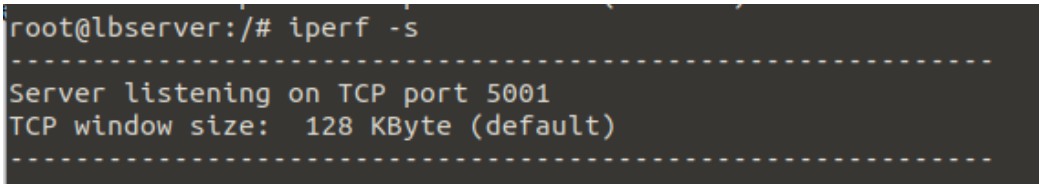
Figura 29: Prueba de acceso al servidor balanceador de carga

Fuente: Elaboración propia.

Adicionalmente instalamos la herramienta Iperf, Hpot para el monitoreo de los procesos y obtener los porcentajes de consumo de los recursos,



Una vez instalado iniciamos el servicio para a la espera de las interacciones de los usuarios.



Instalación y configuración de Frontend (Nodo 1, Nod 2)

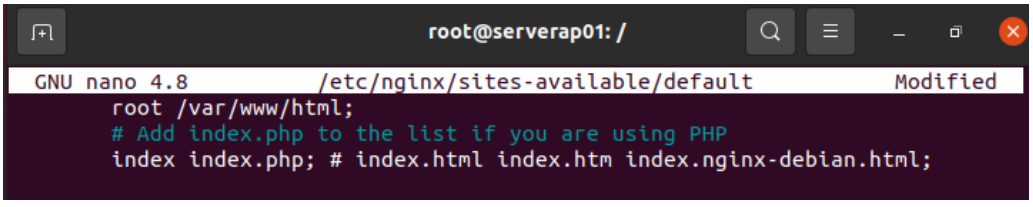
Este proceso de instalación y configuración de los Nodo 1 y Nodo 2 tiene el mismo procedimiento del servidor de balanceo.

Nos conectamos al nodo1 y ejecutamos el comando para actualizar el sistema, se realiza la instalación del servidor web con todas sus dependencias.

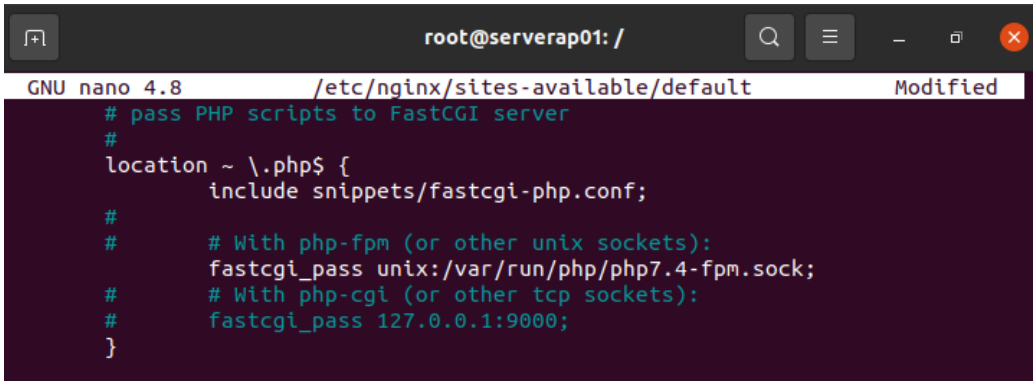
```
~$ sudo apt update
~$ sudo apt install – y nginx
~$ sudo systemctl start nginx
```

Estos dos nodos tendrán la función de servidor de aplicaciones web por lo cual se instalarán el intérprete de *php* con sus dependencias *fpm* y configuramos el acceso al sitio, cambiando el html a php en el archivo default de nginx.

```
~$ sudo apt install php
~$ sudo apt install php-fpm
~$ sudo /nginx/sites-available/default
```



Además, habilitamos el servicio a PHP y los *snippets* para que el contenido de la aplicación se muestre y para terminar validamos y reiniciamos el servicio Nginx.




```
~$ sudo nginx -t
~$ sudo systemctl restart nginx
```

Ahora desplegamos la aplicación web y realizamos la configuración de conexión al servidor de BD.

Figura 30: Despliegue de la aplicación web en el servidor

Fuente: Elaboración Propia

Creación de WebService [Listar]

Para estas pruebas se implementos un servicio web para poder obtener datos del BD para poder ser llamada en cada una de las pruebas con la finalidad de que los resultados sean más completos.

Para ello se implementó un Web Service en lado de servidor el cual muestra listado de proyectos.

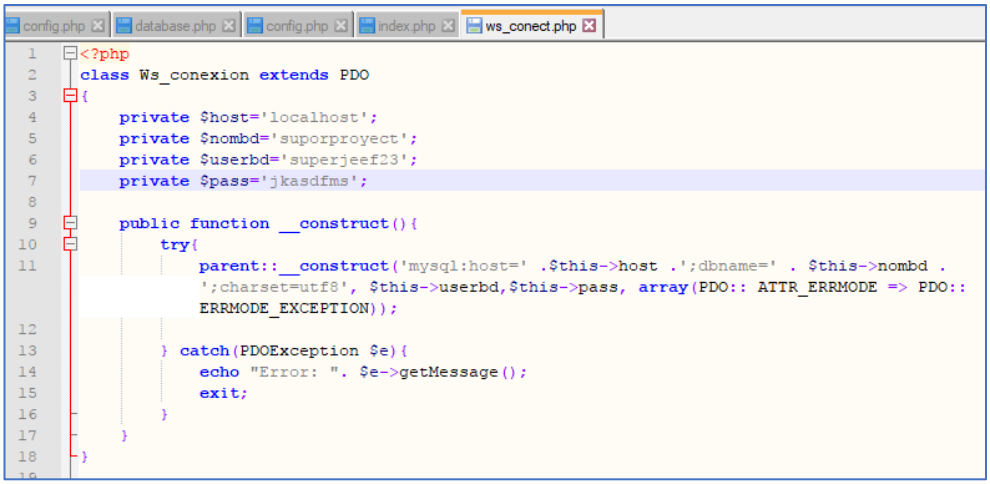


Figura 31: Implementación de la conexión al BD para el Servicio Web

Fuente: Elaboración Propia

Figura 32: Implementación de Web Service, con la consulta de listar datos.

Fuente: Elaboración Propia

Figura 33: Resultado de la ejecución des Web Service.

Fuente: Elaboración Propia

Instalación y configuración de Backend (Nodo 4)

En esta parte realizamos la instalación de aplicación PhpMyadmin para la gestión y creamos de Base de datos de la aplicación web.

Instalación de Servidor Web Nginx

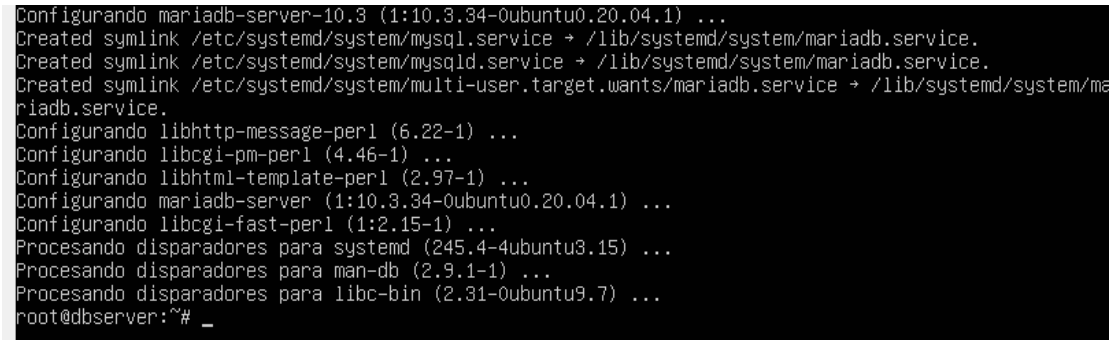
Seguimos el procedimiento de instalación del sistema Linux, ejecutamos los comandos de actualización, instalación del servidor Web y establecer el inicio del servicio.

```
~$ sudo apt update
~$ sudo apt install – y nginx
~$ sudo systemctl start nginx
```

Instalar Gestor de Base de Datos MariaDB

La funcionalidad del Nodo es de servidor de BD, para lo cual procedemos a instalar un sistema de gestión de BD en este caso DBMaria, activamos su arranque automático e iniciamos el servidor.

```
~$ sudo yum install mariadb-server
~$ sudo systemctl enable mariadb
~$ sudo systemctl start mariadb
```



Por seguridad se debe de configurar las reglas de acceso de IP autorizados desde los Nodos 1 y nodo 2 que tienen las aplicaciones que se conectan a obtener información mediante *iptables*. Para fortalecer la seguridad del servidor MariaDB, se ejecuta el comando *mysql_secure_installation* al realizar este proceso nos solicitará información del usuario *root* para lo cual se debe de definir una contraseña segura.

Asu vez creamos un nuevo usuario administrador:

```
$ Sudo mariadb
Mariadb[(none)]>GRANT ALL ON *.* TO 'admin'@'localhost' IDENTIFIED BY
'password' WITH GRANT OPTION;
Mariadb[(none)]> FLUSH PRIVILEGES;
Mariadb[(none)]> Exit;
```

Realizamos la prueba de MariaDB, para lo cual verificamos su estado del servicio.

```
$ sudo systemctl status mariadb
```

```
MariaDB [(none)]> exit
Bye
root@dbserver:~# sudo systemctl status mariadb
● mariadb.service - MariaDB 10.3.34 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2022-06-12 23:18:03 UTC; 9min ago
       Docs: man:mysqld(8)
             https://mariadb.com/kb/en/library/systemd/
   Main PID: 2683 (mysqld)
     Status: "Taking your SQL requests now..."
      Tasks: 31 (limit: 2274)
     Memory: 64.0M
     CGroup: /system.slice/mariadb.service
             └─2683 /usr/sbin/mysqld
```

Para mayor seguridad de su funcionalidad ejecutamos el comando mysqladmin con el nuevo usuario administrador creado anteriormente:

```
$ mysqladmin -u admin -p versión
```

```
Threads: 6  Questions: 78  Slow queries: 0  Opens: 33  Flush tables: 1  Open tables: 26  Queries per
 second avg: 0.083
root@dbserver:~# mysqladmin -u superjeef -p version
Enter password:
mysqladmin  Ver 9.1 Distrib 10.3.34-MariaDB, for debian-linux-gnu on x86_64
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Server version          10.3.34-MariaDB-0ubuntu0.20.04.1
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/run/mysqld/mysqld.sock
Uptime:                 15 min 57 sec

Threads: 6  Questions: 79  Slow queries: 0  Opens: 33  Flush tables: 1  Open tables: 26  Queries per
 second avg: 0.082
root@dbserver:~# _
```

Instalar Aplicación de gestión de Base de Datos

Para gestión de la BD necesitamos una aplicativo como es el phpMyAdmin, para instalar la herramienta, para ello ejecutamos los siguientes comandos.

```
~$ wget https://files.phpmyadmin.net/phpMyAdmin/5.2.0/phpMyAdmin-5.2.0-all-languages.tar.xz
~$ sudo tar xf phpMyAdmin-5.2.0-all-languages.tar.xz -C /var/www/
```

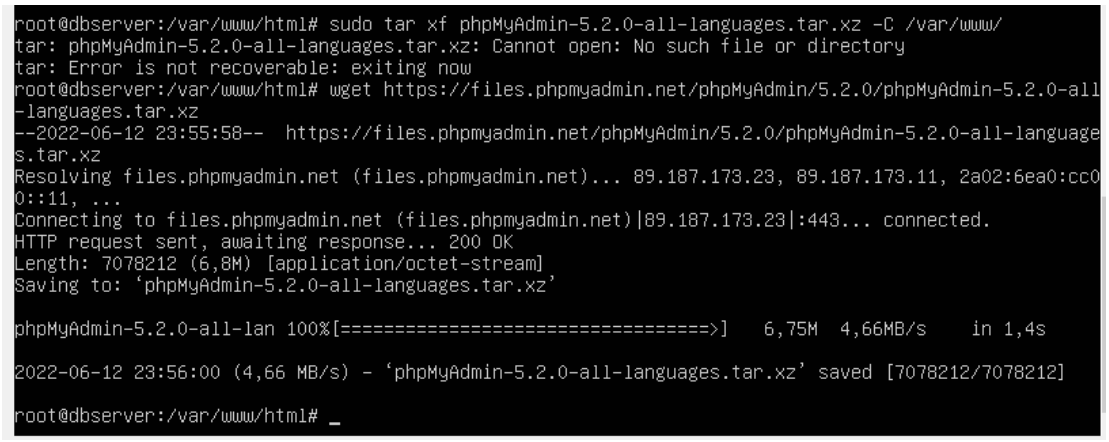
Luego realizamos el cambio de nombre al directorio para una mejor administración.

```
~$ sudo ln -s /var/www/phpMyAdmin-5.2.0-all-languages/ /var/www/phpmyadmin
~$ sudo mkdir /var/www/phpmyadmin/tmp
~$ sudo chown www-data: /var/www/phpmyadmin/tmp/
```

Instalar PHP

Para instalar la extensión para que funciones correctamente el phpmyadmin, ejecutamos los siguientes comandos de instalación y activación del servicio.

```
~$ sudo apt install -y php-bz2 php-mbstring php-xml php-zip
~$ sudo apt install -y php8.0-bz2 php8.0-mbstring php8.0-xml php8.0-zip
~$ sudo systemctl reload apache2
```



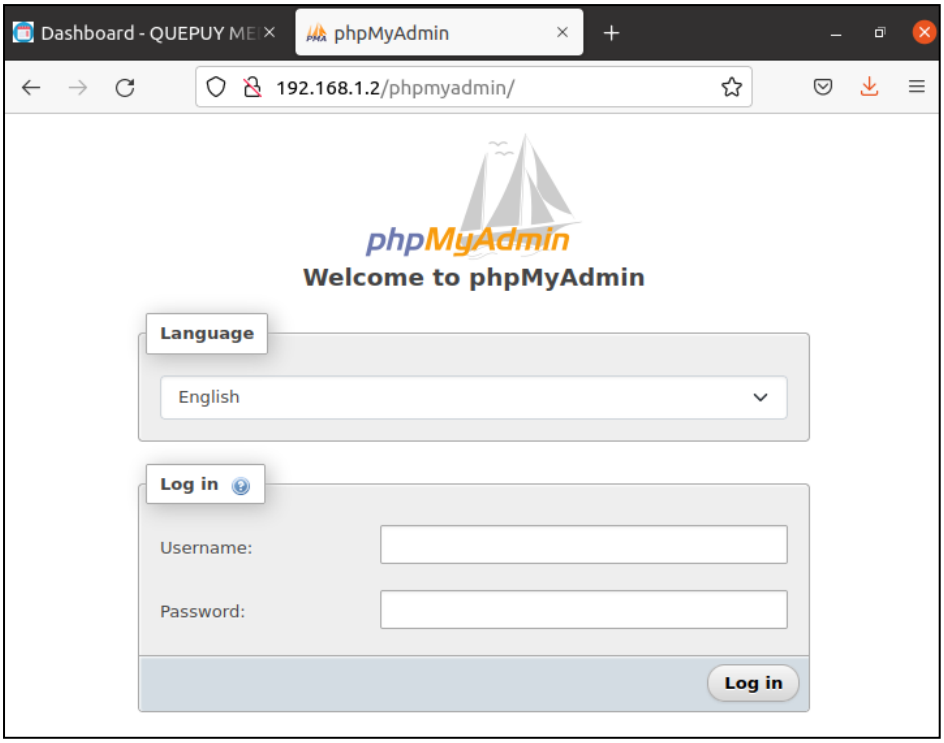


Figura 34: Despliegue del gestor de BD PhpMyadmin

Fuente: Elaboración propia

Obj 3.- Implementar algoritmos balanceadores en la arquitectura de Clúster de Servidor.

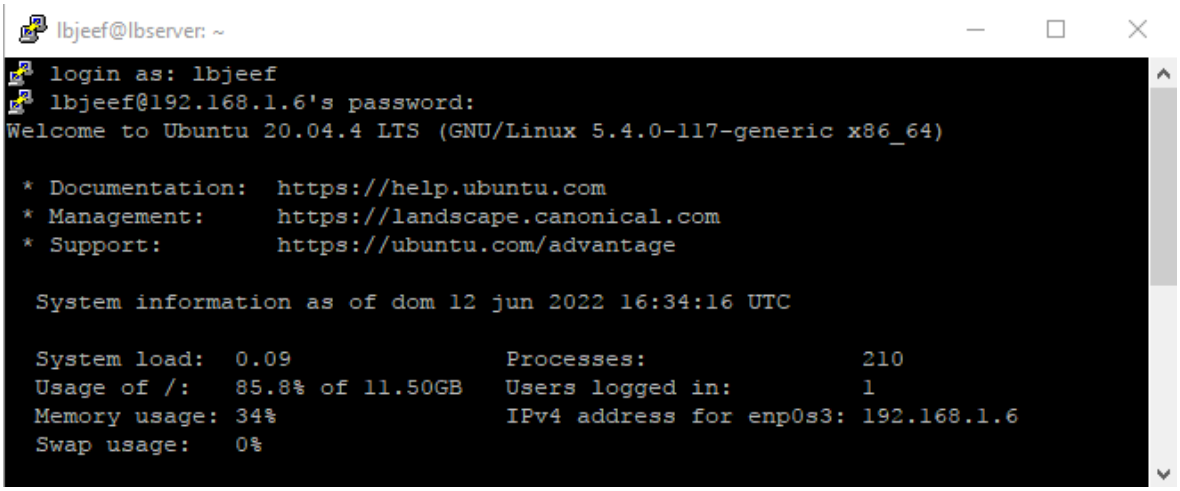
En esta etapa del proyecto, se implementa cada algoritmo aplicando sus parámetros necesarios para mejorar la disponibilidad del servidor.

Configuración del Servidor de Balanceo de carga.

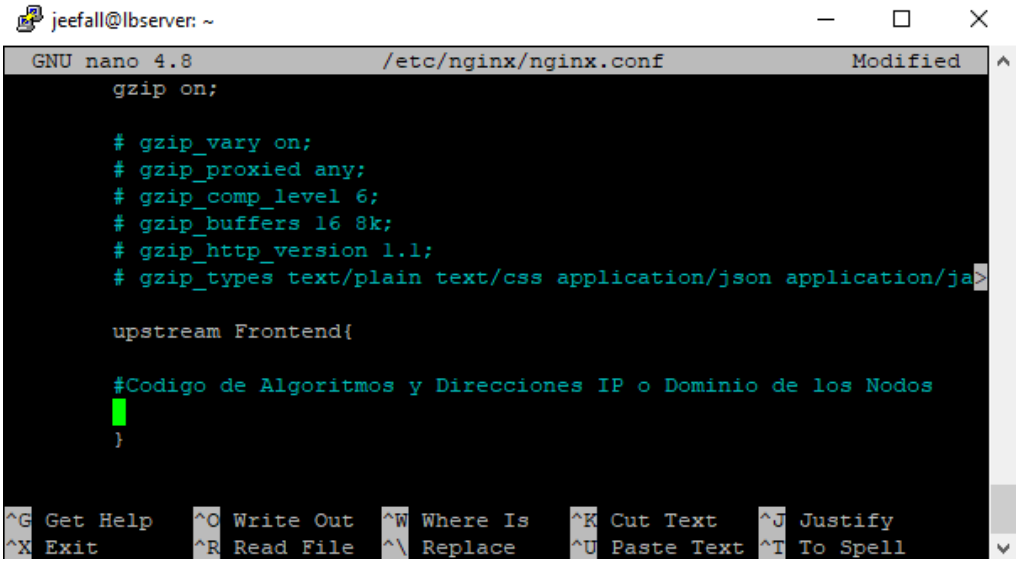
La manera de implementación de los algoritmos varía dependiendo del tipo servidor web que se haya implementado, en este caso el servidor web/proxy inverso es Nginx,

La configuración se realiza en el archivo de configuración http, en el cual se define el algoritmo, las direcciones IP o dominio de los nodos que forman el front-end del clúster y parámetros de control de la seguridad del clúster.

En el cual vamos a configurar a los algoritmos, round robin (RR), Weighted Round Robin (WRR), Least Connection (LC) y Dynamic Consistent Hashing (DCH).



En este archivo se configuran cada uno de los algoritmos, dentro de la función Frontend, en el cual se define el algoritmo a utilizar, asignando parámetros según el tipo de algoritmo para el correcto funcionamiento de balanceo de carga.



Finalmente, la implementación de los algoritmos queda de la siguiente manera.

Round Robin (RR)

```
upstream Frontend {
  server 192.168.1.5;
  server 192.168.1.6;
}

server {
     listen 80;
      location / {
        proxy_pass http://Frontend;
    }
}
```

Weighted Round Robin (WRR)

```
upstream Frontend {
  server 192.168.1.5 weight=1;
  server 192.168.1.6 weight=2;
}

server {
     listen 80;
      location / {
        proxy_pass http://Frontend;
    }
}
```

Least Conection (LC)

```
upstream Frontend {
  least_conn;
  server 192.168.1.5;
  server 192.168.1.6;
}

server {
     listen 80;
      location / {
        proxy_pass http://Frontend;
    }
}
```

Consistent Hashing (CH)

```
upstream Frontend {
  hash $request_uri consistent;
  server 192.168.1.5;
  server 192.168.1.6;
}

server {
     listen 80;
      location / {
        proxy_pass http://Frontend;
    }
}
```

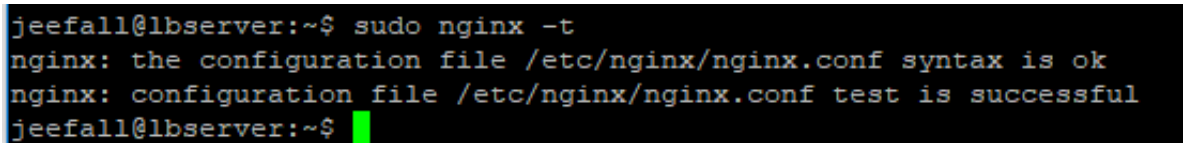
En cuanto al Backend, se debe de configurar de preferencia configurar la directiva Hash como parte de la configuración del grupo de servidores, para manejar las sesiones

Al finalizar de configurar cada uno de los algoritmos, se debe de realizar un test de código, para validar si el script implementado está correcto y luego reiniciar el servicio del servidor web proxy.

Para realizar el test de código.

```
~$ sudo nginx -t
```

En el cual debe dar como resultado que esta correcto de lo contrario indicara que ha fallado el test y muestra el error para corregir.



Reiniciar el servidor Web Proxy.

```
~$ sudo systemcl restart nginx
```

Obj 4.- Realizar las pruebas usando algoritmos balanceadores de carga en un Clúster de Servidores.

En esta etapa final, se realiza la etapa de las pruebas de las estrategias o algoritmos de balanceo de carga, para lo cual se definieron escenarios de prueba en base a la cantidad de usuarios simultáneos en un determinado tiempo y a su vez por el número de peticiones que se enviaran al front-end.

4.1. Rendimiento de clúster de servidor mediante grupos de usuarios concurrentes y un número de peticiones.

Mediante un host cliente, ejecutamos las peticiones de carga de envío de peticiones necesarios para realizar las pruebas de rendimiento de nuestro balanceador de carga, para lo cual tenemos en cuenta la cantidad de usuario concurrentes y el número de peticiones.

Tabla 16: Escenarios de prueba con Apache Benchmark

Escenario Prueba	Número de cliente	Número de Peticiones
01	100	500
02	300	700
03	400	1000

Fuente: Elaboración Propia

Para realizar estas pruebas se realizará mediante el programa APACHE BENCHMARK, para poder enviar las peticiones a nuestro servidor web mediante https o https, para medir el consumo de CPU y Memoria, instalamos la herramienta HPOT para realizar visualizar el visor de procesos interactivo.

4.1.1. Escenario de prueba 01

Estas pruebas de carga se realizan mediante el Apache Benchmark comando y a su vez mediante la aplicación Htop se captura el consumo de CPU y Memoria.

Prueba de Algoritmo Round Robin (RR), con 100 usuario concurrentes y 500 solicitudes que se envían desde un cliente al servidor de balanceo de carga.

```
# ab -g test01RR.csv -c 100 -n 500 -f ALL http://192.168.1.6/gestionproyadm/ws
```

Una vez ejecutada se muestran los resultados como el tiempo utilizado para la respuesta de la prueba, el tiempo de respuesta, los datos transferidos. Datos que se extraerán para su posterior análisis.

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      100
Time taken for tests:   0.224 seconds
Complete requests:      500
Failed requests:        0
Non-2xx responses:      500
Total transferred:      198000 bytes
HTML transferred:       89000 bytes
Requests per second:    2230.78 [#/sec] (mean)
Time per request:       44.827 [ms] (mean)
Time per request:       0.448 [ms] (mean, across all concurrent requests)
Transfer rate:          862.68 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   10   5.4      9      25
Processing:    11   31   9.7     30      56
Waiting:        1   24   7.0     22      44
Total:         11   41   8.7     43      61
```

Figura 35. Resultado de prueba de carga.

Consumo de recursos de servidor

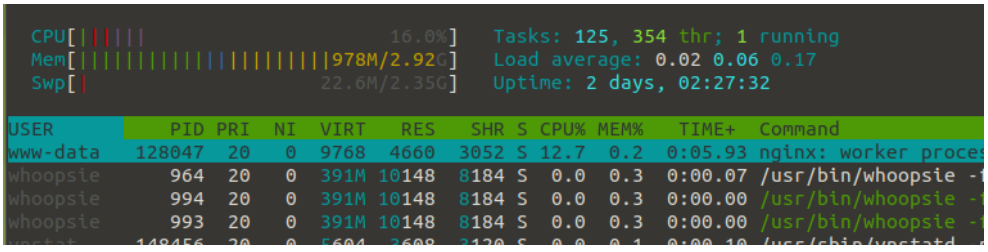


Figura 36. Resultado de consumo de CPU y RAM.

Prueba de Algoritmo Weight Round Robin (WRR), con 100 usuario concurrentes y 500 solicitudes que se envían desde un cliente al servidor de balanceo de carga.

```
# ab -g test01WRR.csv -c 100 -n 500 -f ALL http://192.168.1.6/gestionproyadm/ws
```

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      100
Time taken for tests:   0.259 seconds
Complete requests:      500
Failed requests:        0
Non-2xx responses:      500
Total transferred:      198000 bytes
HTML transferred:       89000 bytes
Requests per second:    1927.23 [#/sec] (mean)
Time per request:       51.888 [ms] (mean)
Time per request:       0.519 [ms] (mean, across all concurrent requests)
Transfer rate:          745.30 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   14   5.9     14      26
Processing:     4   35  11.5     36      58
Waiting:        1   28   9.5     26      52
Total:         12   49   8.7     50      63
```

Consumo recursos

```
  CPU[|||||||                    17.1%]   Tasks: 126, 355 thr; 1 running
  Mem[||||||||||||||||||||976M/2.92G]   Load average: 0.05 0.06 0.12
  Swp[|                   22.6M/2.35G]   Uptime: 2 days, 02:34:02

USER       PID PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
www-data 159344  20   0  9716  4480  2960 S 13.2  0.1  0:00.20 nginx: worker proces
whoopsie    964  20   0  391M 10148  8184 S  0.0  0.3  0:00.07 /usr/bin/whoopsie -f
whoopsie    994  20   0  391M 10148  8184 S  0.0  0.3  0:00.00 /usr/bin/whoopsie -f
```

Prueba de Algoritmo Least Conecction (LC), con 100 usuario concurrentes y 500 solicitudes que se envían desde un cliente al servidor de balanceo de carga.

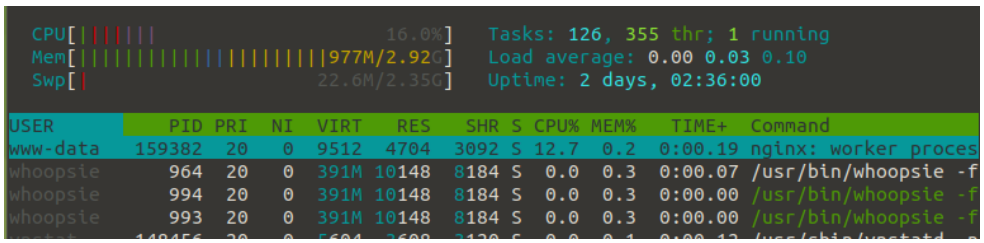
```
# ab -g test01LC.csv -c 10 -n 100 -f ALL http://192.168.1.6/gestionproyadm/ws
```

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      100
Time taken for tests:   0.243 seconds
Complete requests:      500
Failed requests:        0
Non-2xx responses:      500
Total transferred:      198000 bytes
HTML transferred:       89000 bytes
Requests per second:    2053.75 [#/sec] (mean)
Time per request:       48.691 [ms] (mean)
Time per request:       0.487 [ms] (mean, across all concurrent requests)
Transfer rate:          794.22 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   10   6.0      10      26
Processing:     3   36  12.0      38      60
Waiting:        2   29  10.8      25      54
Total:         12   46  10.2      48      68
```

Consumo de recursos



Prueba de Algoritmo Hash Consistente (HC), con 10 usuario concurrentes y 100 solicitudes que se envían desde un cliente al servidor de balanceo de carga.

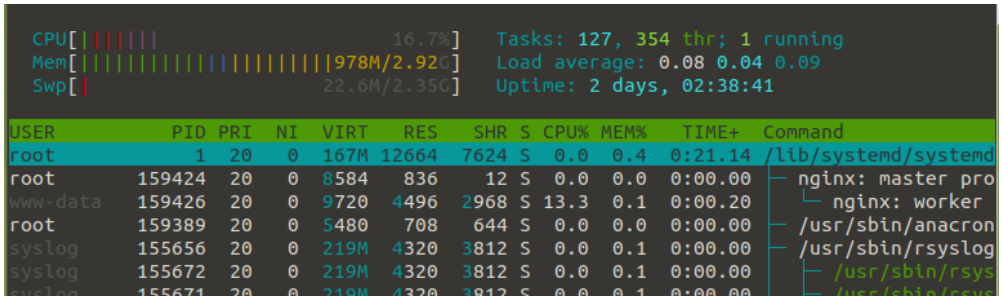
```
# ab -g test01HC.csv -c 10 -n 100 -f ALL http://192.168.1.6/gestionproyadm/ws
```

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      100
Time taken for tests:   0.237 seconds
Complete requests:      500
Failed requests:        0
Non-2xx responses:      500
Total transferred:      198000 bytes
HTML transferred:       89000 bytes
Requests per second:    2106.85 [#/sec] (mean)
Time per request:       47.464 [ms] (mean)
Time per request:       0.475 [ms] (mean, across all concurrent requests)
Transfer rate:          814.76 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    2   3.0      0      11
Processing:    12   41   6.8     44      50
Waiting:        1   41   7.1     44      50
Total:         12   43   4.7     44      54
```

Consumo de recursos

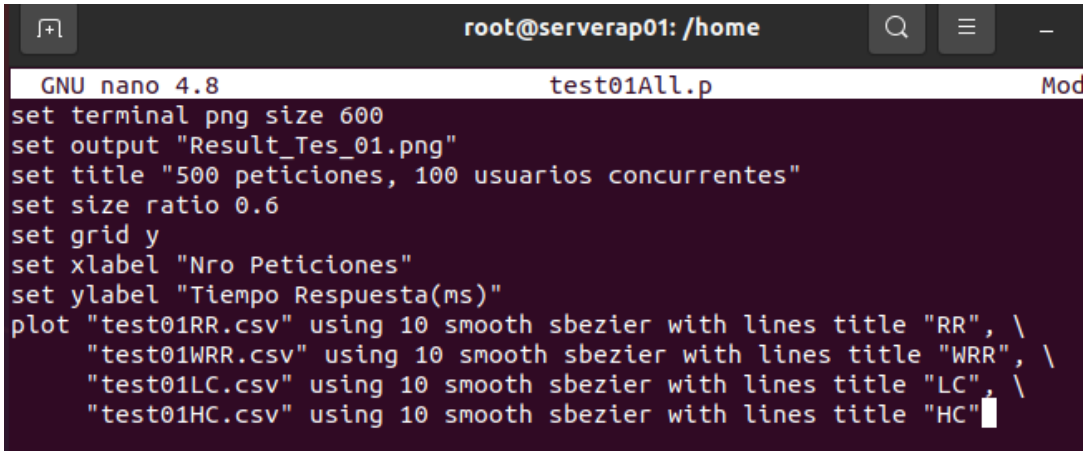


Posteriormente creamos el archivo **test01Aall.p** para representar el grafico de las pruebas de los algoritmos e balanceo de carga ejecutados RR,WRR,

LC, y HC

```
# nano test01All.p
```

Ingresos el script para la generación del gráfico e indicamos que los datos a utilizar son los resultados guardados en archivos csv, de las pruebas antes ejecutadas.



Una vez creado el archivo usamos el comando *gnuplot* para generar el grafico.

```
# gnuplot test01All.p
```

4.1.2. Escenario de prueba 02

Para nuestro segundo escenario tamos como referencia 300 usuario concurrentes y 700 peticiones.

Algortimos Round Robin (RR)

ab -g test02RR.csv -c 300 -n 500 -f ALL http://192.168.1.6/gestionproyadm/ws

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      300
Time taken for tests:   0.336 seconds
Complete requests:      700
Failed requests:        0
Non-2xx responses:      700
Total transferred:      277200 bytes
HTML transferred:       124600 bytes
Requests per second:    2084.61 [#/sec] (mean)
Time per request:       143.912 [ms] (mean)
Time per request:       0.480 [ms] (mean, across all concurrent requests)
Transfer rate:          806.16 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   12  11.8      5      32
Processing:    17  102  18.3    105     141
Waiting:        1  101  18.3    105     140
Total:         33  114  15.8    111     155
```

Consumo recursos

```
  CPU[||||||||||||                  22.7%]   Tasks: 125, 356 thr; 1 running
  Mem[||||||||||||||||||||||984M/2.92G]   Load average: 0.20 0.05 0.02
  Swp[|                     22.6M/2.35G]   Uptime: 2 days, 03:23:00

USER        PID PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
www-data 159451  20   0 11300  6456  3056 S 20.0  0.2  0:01.02 nginx: worker proces
whoopsie    964  20   0  391M 10148  8184 S  0.0  0.3  0:00.07 /usr/bin/whoopsie -f
whoopsie    994  20   0  391M 10148  8184 S  0.0  0.3  0:00.00 /usr/bin/whoopsie -f
whoopsie    993  20   0  391M 10148  8184 S  0.0  0.3  0:00.00 /usr/bin/whoopsie -f
vnstat   148456  20   0  5604  3608  3120 D  0.0  0.1  0:00.28 /usr/sbin/vnstatd -n
```

Prueba de Algoritmo Weight Round Robin (WRR)

ab -g test02WRR.csv -c 300 -n 500 -f ALL http://192.168.1.6/gestionproyadm/ws

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      300
Time taken for tests:   0.362 seconds
Complete requests:      700
Failed requests:        0
Non-2xx responses:      700
Total transferred:      277200 bytes
HTML transferred:       124600 bytes
Requests per second:    1935.88 [#/sec] (mean)
Time per request:       154.968 [ms] (mean)
Time per request:       0.517 [ms] (mean, across all concurrent request
Transfer rate:          748.64 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   30  18.5     35      69
Processing:    29  109  46.4    121     176
Waiting:        2   98  40.0    111     154
Total:         39  138  32.7    150     188
```

Consumo recursos

```
  CPU[||||||||                    24.0%]   Tasks: 127, 354 thr; 1 running
  Mem[||||||||||||||||||||985M/2.92G]   Load average: 0.12 0.05 0.01
  Swp[|                     22.6M/2.35G]   Uptime: 2 days, 03:24:28

USER        PID PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
www-data 159643  20   0  9632  4692  3092 S 19.3  0.2  0:00.29 nginx: worker pro
whoopsie    964  20   0  391M 10148  8184 S  0.0  0.3  0:00.07 /usr/bin/whoopsie
whoopsie    994  20   0  391M 10148  8184 S  0.0  0.3  0:00.00 /usr/bin/whoopsie
whoopsie    993  20   0  391M 10148  8184 S  0.0  0.3  0:00.00 /usr/bin/whoopsie
```

Prueba de Algoritmo Least Conecction (LC)

ab -g test02LC.csv -c 300 -n 700 -f ALL http://192.168.1.6/gestionproyadm/ws

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      300
Time taken for tests:   0.348 seconds
Complete requests:      700
Failed requests:        0
Non-2xx responses:      700
Total transferred:      277200 bytes
HTML transferred:       124600 bytes
Requests per second:    2011.95 [#/sec] (mean)
Time per request:       149.109 [ms] (mean)
Time per request:       0.497 [ms] (mean, across all concurrent requests)
Transfer rate:          778.06 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   30  17.7     32      69
Processing:    21  102  40.5    106     168
Waiting:        3   92  34.6     94     153
Total:         33  131  29.9    142     182
```

Consumo Recursos

```
  CPU[|||||||                    23.0%]   Tasks: 128, 354 thr; 1 running
  Mem[||||||||||||||||||||987M/2.92G]   Load average: 0.14 0.06 0.01
  Swp[|                    22.6M/2.35G]   Uptime: 2 days, 03:26:08

USER       PID PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
www-data 159674  20   0 10224  5356  3032 S 20.3  0.2  0:00.30 nginx: worker proces
whoopsie    994  20   0  391M 10148  8184 S  0.0  0.3  0:00.00 /usr/bin/whoopsie -f
whoopsie    993  20   0  391M 10148  8184 S  0.0  0.3  0:00.00 /usr/bin/whoopsie -f
whoopsie    964  20   0  391M 10148  8184 S  0.0  0.3  0:00.07 /usr/bin/whoopsie -f
```

Prueba de Algoritmos Hash Consistente (HC)

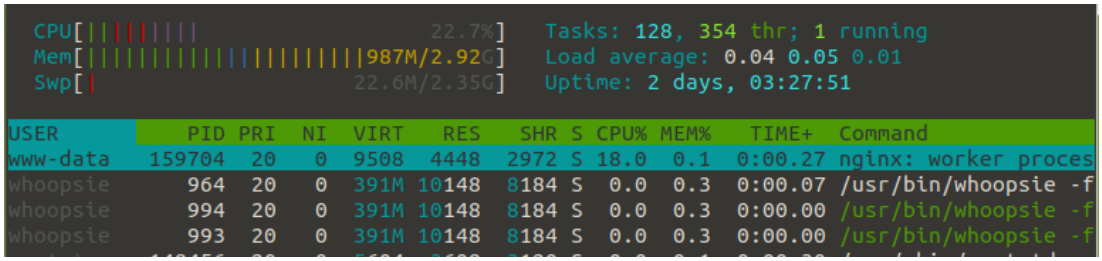
ab -g test02HC.csv -c 300 -n 700 -f ALL http://192.168.1.6/gestionproyadm/ws

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      300
Time taken for tests:   0.336 seconds
Complete requests:      700
Failed requests:        0
Non-2xx responses:      700
Total transferred:      277200 bytes
HTML transferred:       124600 bytes
Requests per second:    2084.43 [#/sec] (mean)
Time per request:       143.925 [ms] (mean)
Time per request:       0.480 [ms] (mean, across all concurrent request
Transfer rate:          806.09 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   14  12.0     10      40
Processing:    25   97  16.9     95     143
Waiting:        2   96  16.2     95     136
Total:         43  111  16.8    106     155
```

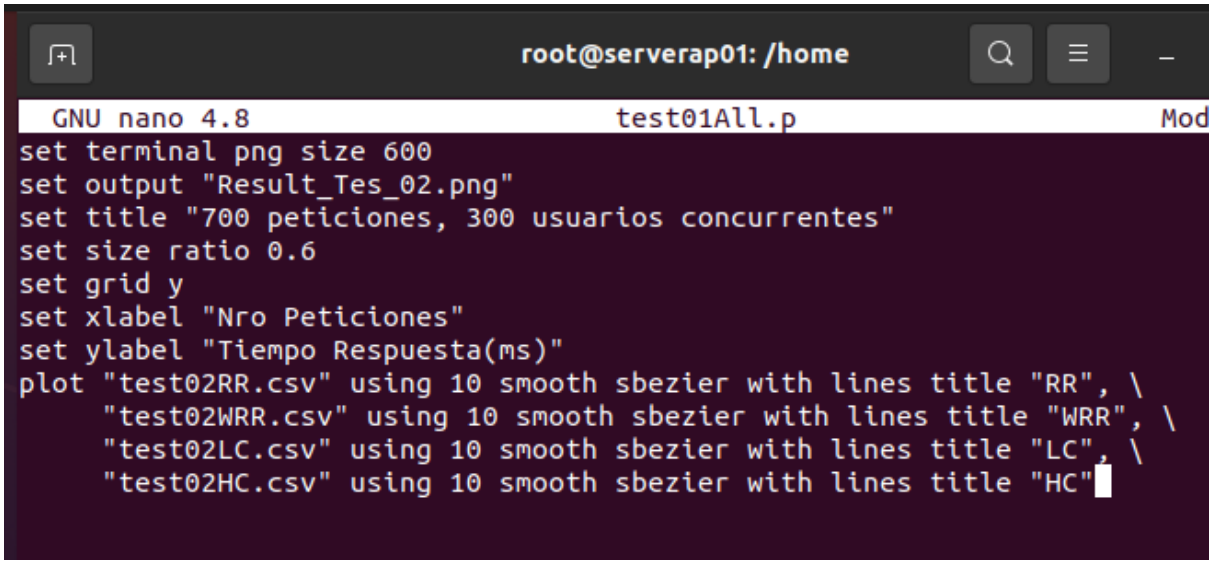
Consumo de Recursos



Para generar la gráfica creamos el archivo **test02Aall.p** para representar el los datos de las pruebas de los algoritmos de balanceo de carga ejecutados RR,WRR, LC, y HC

```
# nano test02All.p
```

Ingresos el script para la generación del gráfico y indicamos que los datos a utilizar son los resultados guardados en archivos csv, de las pruebas antes ejecutadas.



Una vez creado el archivo usamos el comando *gnuplot* para generar el grafico.

```
# gnuplot test02All.p
```

4.1.3. Escenario de prueba 03

Para nuestro tercer escenario, elevamos nuestros valores de usuarios concurrentes y las peticiones a 50 usuario concurrentes y 500 peticiones.

Prueba de Algoritmo Round Robin (RR)

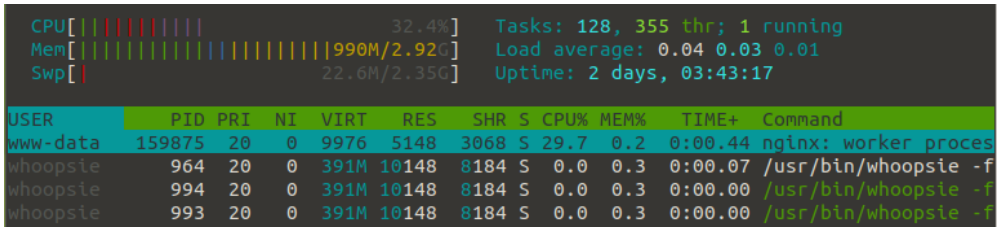
ab -g test02RR.csv -c 1000 -n 400 -f ALL http://192.168.1.6/gestionproyadm/ws

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      400
Time taken for tests:   0.508 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      396000 bytes
HTML transferred:       178000 bytes
Requests per second:    1967.91 [#/sec] (mean)
Time per request:       203.262 [ms] (mean)
Time per request:       0.508 [ms] (mean, across all concurrent requests)
Transfer rate:          761.03 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   38  24.4     39      91
Processing:    24  144  60.1    164     245
Waiting:        1  128  50.9    133     221
Total:         47  183  43.2    200     246
```

Consumo recursos



Prueba de Algoritmo Weight Round Robin (WRR)

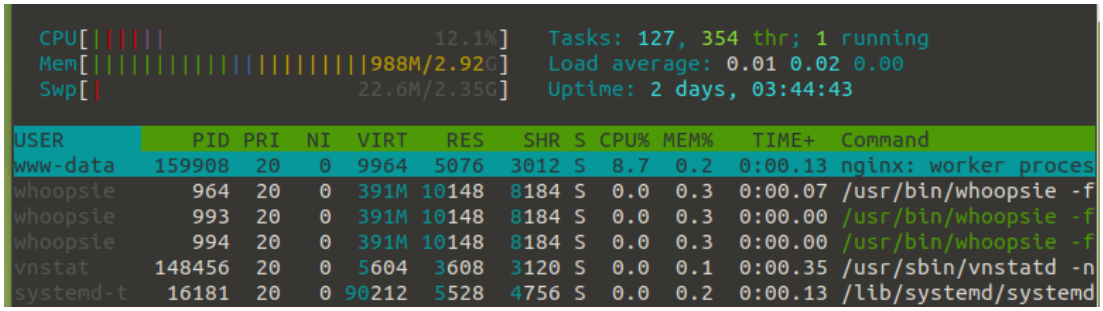
ab -g test02WRR.csv -c 1000 -n 400 -f ALL http://192.168.1.6/gestionproyadm/ws

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      400
Time taken for tests:   0.508 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      396000 bytes
HTML transferred:       178000 bytes
Requests per second:    1967.91 [#/sec] (mean)
Time per request:       203.262 [ms] (mean)
Time per request:       0.508 [ms] (mean, across all concurrent request
Transfer rate:          761.03 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   38  24.4     39      91
Processing:    24  144  60.1    164     245
Waiting:        1  128  50.9    133     221
Total:         47  183  43.2    200     246
```

Consumo recursos



Prueba de Algoritmo Least Conecction (LC)

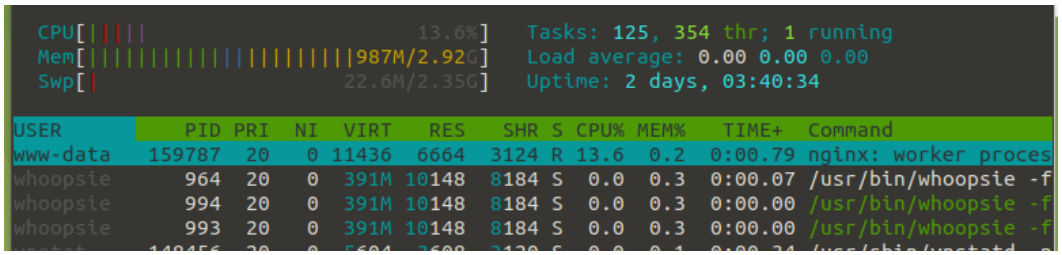
ab -g test03LC.csv -c 1000 -n 400 -f ALL http://192.168.1.6/gestionproyadm/ws

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      400
Time taken for tests:   0.500 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      396000 bytes
HTML transferred:       178000 bytes
Requests per second:    2001.02 [#/sec] (mean)
Time per request:       199.898 [ms] (mean)
Time per request:       0.500 [ms] (mean, across all concurrent requests)
Transfer rate:          773.83 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   29  14.1     33      59
Processing:    20  143  37.7    149     214
Waiting:        1  131  31.3    135     180
Total:         49  172  30.7    175     234
```

Consumo Recursos



Prueba de Algoritmos Hash Consistente (HC)

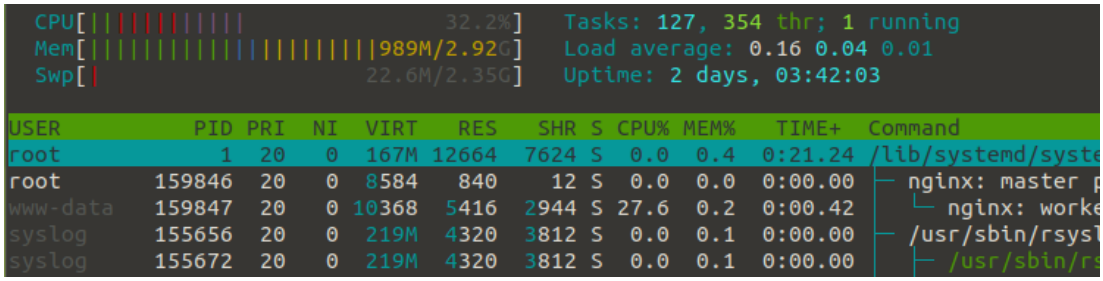
ab -g test03HC.csv -c 1000 -n 400 -f ALL http://192.168.1.6/gestionproyadm/ws

```
Document Path:          /gestionproyadm/ws
Document Length:        178 bytes

Concurrency Level:      400
Time taken for tests:   0.482 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      396000 bytes
HTML transferred:       178000 bytes
Requests per second:    2073.57 [#/sec] (mean)
Time per request:       192.904 [ms] (mean)
Time per request:       0.482 [ms] (mean, across all concurrent requests)
Transfer rate:          801.89 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   22  14.5     23      50
Processing:    21  131  29.9    130     191
Waiting:        1  128  27.7    128     186
Total:         46  153  23.7    147     210
```

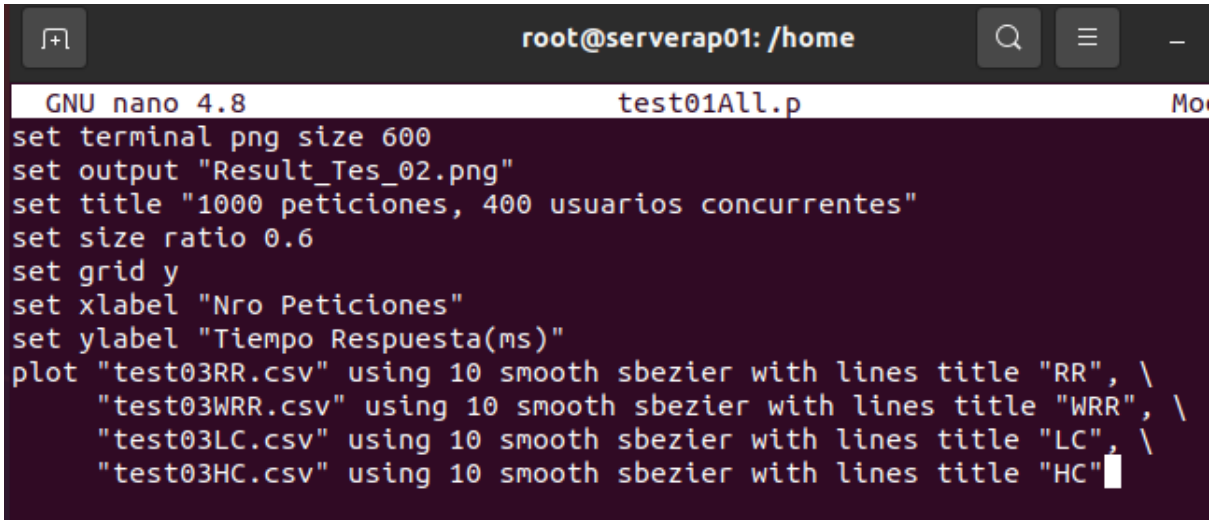
Consumo de Recursos



Para generar la gráfica creamos el archivo **test03Aall.p** para representar el los datos de las pruebas de los algoritmos de balanceo de carga ejecutados RR,WRR, LC, y HC

```
# nano test03All.p
```

Ingresos el script para la generación del gráfico y indicamos que los datos a utilizar son los resultados guardados en archivos csv, de las pruebas antes ejecutadas.



Una vez creado el archivo usamos el comando *gnuplot* para generar el grafico.

```
# gnuplot test03All.p
```

Pruebas de rendimiento con Apache Jmeter

Para realizar esta prueba se ejecutarán mediante la herramienta APACHE JMETER, para iniciar se definen dos escenarios, tal como se describe en la tabla N°. 16

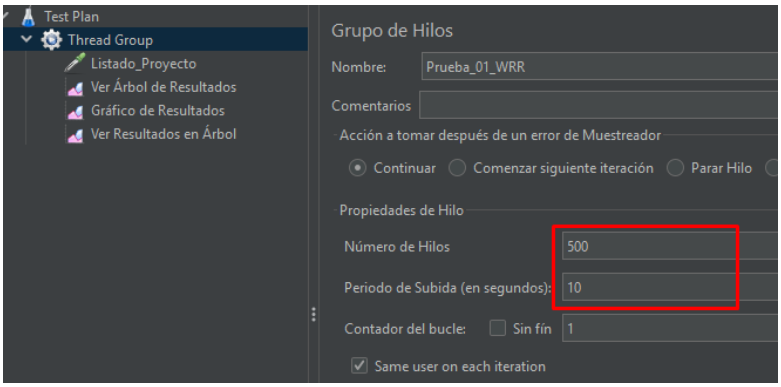
Tabla 17: Escenarios de prueba para Apache Jmeter

Escenario Prueba	Número de cliente	Peticiones	Intervalo
01	500	50	10(s)
02	1000	100	10(s)

Prueba rendimiento 01 con Apache Jmeter

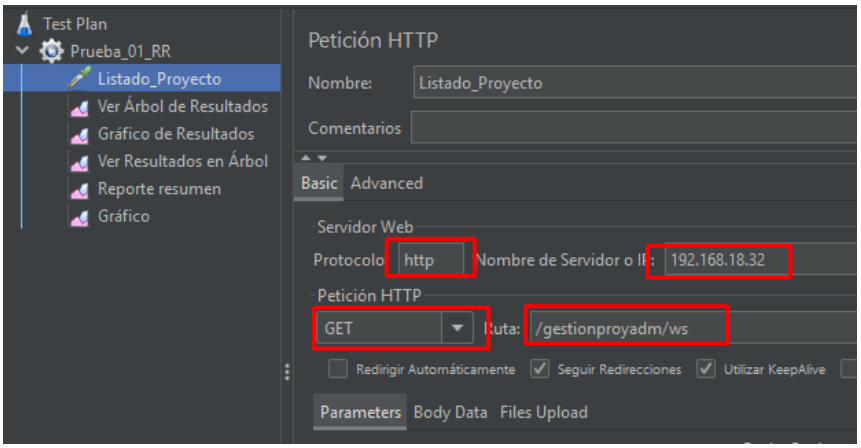
a. Definir grupos de hilos

Se configuro el primer escenario de 500 usuarios, con 50 peticiones cada 10 segundos.



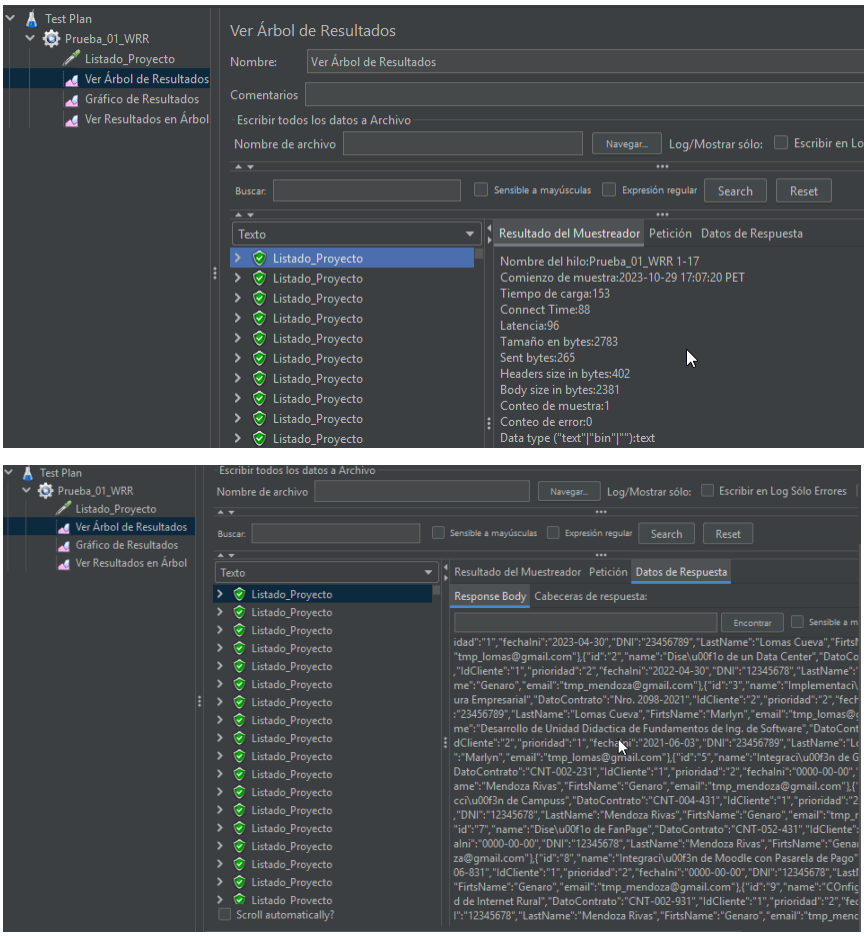
b. Definir los parámetros de petición HTTP

Definimos los datos del Servidor Balanceador de Carga, se indica el protocolo, la dirección de N° IP, el método y la ruta, en este caso se llama a un Web Service, de listado de datos.



c. Agregar El árbol de resultados

En este apartado se muestran el resumen de cada hilo de ejecución, en el cual se observa el estado, asimismo podemos validar la respuesta de la petición como son los datos de respuesta.



d. Reporte Resumen

En este espacio se observa el resumen de la prueba ejecutada, obteniendo datos como, la muestra, el rendimiento, la latencia, error entre otros datos.

Etiqueta	# Muestr...	Media ↑	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec
Listado_Proyecto	1000	22984	0	60058	29170.21	60.40%	15.6/sec	39.06	1.60
Total	1000	22984	0	60058	29170.21	60.40%	15.6/sec	39.06	1.60

Ejecutar escenario 01: de 500 usuarios, ejecutando 50 peticiones cada 10 segundos

Algoritmo RR, se muestran el resumen

Etiqueta	# Muestr...	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec
Listado_Proyecto	500	126	0	60044	2682.30	0.00%	8.3/sec	22.62
Total	500	126	0	60044	2682.30	0.00%	8.3/sec	22.62

Algoritmos WRR, se muestran el resumen y el árbol de resultados

Etiqueta	# Muestr...	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec
Listado_Proyecto	500	30007	0	60024	30000.97	0.00%	7.1/sec	19.42	1.85
Total	500	30007	0	60024	30000.97	0.00%	7.1/sec	19.42	1.85

Muestra #	Tiempo de com...	Nombre del hilo	Etiqueta	Tiempo de Muest...	Estado	Bytes	Sent Bytes	Latency
1	17:48:12.692	Prueba_01_WRR...	Listado_Proyecto	4		2783	265	2
2	17:48:12.733	Prueba_01_WRR...	Listado_Proyecto	4		2783	265	1
3	17:48:12.772	Prueba_01_WRR...	Listado_Proyecto	12		2783	265	2
4	17:48:12.814	Prueba_01_WRR...	Listado_Proyecto	5		2783	265	2
5	17:48:12.852	Prueba_01_WRR...	Listado_Proyecto	11		2783	265	4
6	17:48:12.894	Prueba_01_WRR...	Listado_Proyecto	5		2783	265	2
7	17:48:12.932	Prueba_01_WRR...	Listado_Proyecto	4		2783	265	2
8	17:48:12.973	Prueba_01_WRR...	Listado_Proyecto	13		2783	265	4
9	17:48:13.013	Prueba_01_WRR...	Listado_Proyecto	20		2783	265	5
10	17:48:13.053	Prueba_01_WRR...	Listado_Proyecto	4		2783	265	1
11	17:48:13.092	Prueba_01_WRR...	Listado_Proyecto	5		2783	265	2
12	17:48:13.132	Prueba_01_WRR...	Listado_Proyecto	18		2783	265	5

Algoritmos LC, se muestran el resumen y el árbol de resultados

Etiqueta	# Muestr...	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec
Listado_Proyecto	500	248	0	120537	5384.85	0.00%	4.1/sec	11.27	1.07
Total	500	248	0	120537	5384.85	0.00%	4.1/sec	11.27	1.07

Algoritmos HC, se muestran el resumen y el árbol de resultados

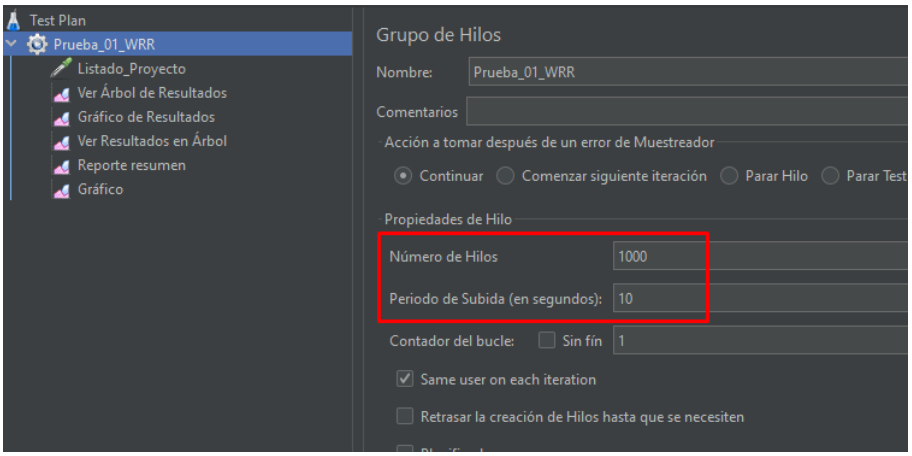
Etiqueta	# Muestr...	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec
Listado_Proyecto	500	128	0	60056	2682.75	0.00%	8.3/sec	22.63	2.15
Total	500	128	0	60056	2682.75	0.00%	8.3/sec	22.63	2.15

Muestra #	Tiempo de com...	Nombre del hilo	Etiqueta	Tiempo de Muest...	Estado	Bytes	Sent Bytes	Latency
1	17:44:35.341	Prueba_01_HC ...	Listado_Proyecto	3		2783	265	1
2	17:44:35.361	Prueba_01_HC ...	Listado_Proyecto	3		2783	265	1
3	17:44:35.381	Prueba_01_HC ...	Listado_Proyecto	4		2783	265	1
4	17:44:35.400	Prueba_01_HC ...	Listado_Proyecto	4		2783	265	2
5	17:44:35.419	Prueba_01_HC ...	Listado_Proyecto	4		2783	265	2
6	17:44:35.440	Prueba_01_HC ...	Listado_Proyecto	15		2783	265	2
7	17:44:35.460	Prueba_01_HC ...	Listado_Proyecto	3		2783	265	1
8	17:44:35.480	Prueba_01_HC ...	Listado_Proyecto	4		2783	265	1
9	17:44:35.500	Prueba_01_HC ...	Listado_Proyecto	4		2783	265	2
10	17:44:35.521	Prueba_01_HC ...	Listado_Proyecto	4		2783	265	1
11	17:44:35.541	Prueba_01_HC ...	Listado_Proyecto	6		2783	265	2
12	17:44:35.560	Prueba_01_HC ...	Listado_Proyecto	10		2783	265	4

Prueba rendimiento 02 con Apache Jmeter

a. Definir grupos de hilos

Se configuro el segundo escenario de 1000 usuarios, con 100 peticiones cada 10 segundos.



Los otras opciones para los resultados se mantiene.

Escenario 02: de 1000 usuarios, ejecutando 100 peticiones cada 10 segundos

Algoritmo RR, se muestran el resumen y el árbol de resultados

Etiqueta	# Muestr...	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec
Listado_Proyecto	1000	22983	0	60034	29169.97	61.20%	15.7/sec	38.94	1.57
Total	1000	22983	0	60034	29169.97	61.20%	15.7/sec	38.94	1.57

Muestra #	Tiempo de com...	Nombre del hilo	Etiqueta	Tiempo de Muest...	Estado	Bytes	Sent Bytes	Latency
690	17:56:15.894	Prueba_01_RR 1...	Listado_Proyecto	60006		2783	265	4
691	17:56:15.904	Prueba_01_RR 1...	Listado_Proyecto	60007		2783	265	4
692	17:56:15.916	Prueba_01_RR 1...	Listado_Proyecto	60006		2783	265	4
693	17:56:15.925	Prueba_01_RR 1...	Listado_Proyecto	60005		2783	265	3
694	17:56:15.933	Prueba_01_RR 1...	Listado_Proyecto	60009		2783	265	6
695	17:56:15.944	Prueba_01_RR 1...	Listado_Proyecto	60009		2783	265	3
696	17:56:15.954	Prueba_01_RR 1...	Listado_Proyecto	60010		2783	265	1
697	17:56:15.964	Prueba_01_RR 1...	Listado_Proyecto	60012		2783	265	4
698	17:56:15.976	Prueba_01_RR 1...	Listado_Proyecto	60013		2783	265	4

Algoritmos WRR, se muestran el resumen y el árbol de resultados

Etiqueta	# Muestr...	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec
Listado_Proyecto	1000	66	0	60066	1898.30	0.00%	16.6/sec	45.25	4.31
Total	1000	66	0	60066	1898.30	0.00%	16.6/sec	45.25	4.31

Muestra #	Tiempo de com...	Nombre del hilo	Etiqueta	Tiempo de Muest...	Estado	Bytes	Sent Bytes	Latency
943	17:53:23.732	Prueba_01_WRR...	Listado_Proyecto	3		2783	265	1
944	17:53:23.743	Prueba_01_WRR...	Listado_Proyecto	3		2783	265	1
945	17:53:23.753	Prueba_01_WRR...	Listado_Proyecto	3		2783	265	1
946	17:53:23.762	Prueba_01_WRR...	Listado_Proyecto	4		2783	265	1
947	17:53:23.772	Prueba_01_WRR...	Listado_Proyecto	4		2783	265	1
948	17:53:23.782	Prueba_01_WRR...	Listado_Proyecto	3		2783	265	1
949	17:53:23.792	Prueba_01_WRR...	Listado_Proyecto	8		2783	265	6
950	17:53:23.802	Prueba_01_WRR...	Listado_Proyecto	2		2783	265	1

Algoritmos LC, se muestran el resumen y el árbol de resultados

Etiqueta	# Muestr...	Media ↑	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec
Listado_Proyecto	1000	66	0	60034	1897.29	0.00%	16.7/sec	45.27	4.31
Total	1000	66	0	60034	1897.29	0.00%	16.7/sec	45.27	4.31

Muestra #	Tiempo de com...	Nombre del hilo	Etiqueta	Tiempo de Muest...	Estado	Bytes	Sent Bytes	Latency
212	18:08:17.441	Prueba_01_HC ...	Listado_Proyecto	3		2783	265	1
213	18:08:17.432	Prueba_01_HC ...	Listado_Proyecto	4		2783	265	1
214	18:08:17.451	Prueba_01_HC ...	Listado_Proyecto	4		2783	265	2
215	18:08:17.461	Prueba_01_HC ...	Listado_Proyecto	4		2783	265	2
216	18:08:17.471	Prueba_01_HC ...	Listado_Proyecto	22		2783	265	14
217	18:08:17.482	Prueba_01_HC ...	Listado_Proyecto	13		2783	265	5
218	18:08:17.492	Prueba_01_HC ...	Listado_Proyecto	7		2783	265	2
219	18:08:17.500	Prueba_01_HC ...	Listado_Proyecto	13		2783	265	2
220	18:08:17.510	Prueba_01_HC ...	Listado_Proyecto	6		2783	265	4
221	18:08:17.521	Prueba_01_HC ...	Listado_Proyecto	3		2783	265	1

Algoritmos HC, se muestran el resumen y el árbol de resultados

Etiqueta	# Muestr...	Media ↑	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec
Listado_Proyecto	1000	22984	0	60032	29170.75	61.70%	15.7/sec	39.62	1.55
Total	1000	22984	0	60032	29170.75	61.70%	15.7/sec	39.62	1.55

IV. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones.

Después de realizar una exhaustiva revisión sistemática de los artículos, se seleccionaron los algoritmos balanceadores de carga, más utilizados y de mejor desempeño en tiempo de respuesta y disponibilidad de un servidor: Round Robin (RR), Weighted Round Robin (WRR), Least Conection (LC) y Consistent Hashing (DCH)

Se determinó la arquitectura de Clúster de servidores para realizar las comparaciones de los algoritmos balanceadores de carga seleccionados, tal esquema está referenciado en los diferentes modelos (hardware y software) de aplicación encontrados en la investigación realizada, lo cual se pudo realizar la implementación definiendo un Back-end para el BD y un Front-end para la aplicación web y el Load Balancer.

Se logró de manera exitosa la implementación de algoritmos balanceadores en la arquitectura de Clúster de Servidor, mediante la configuración del Servidor de Balanceo de carga, así como de los algoritmos seleccionados, logrando validarlo ejecutando un test de funcionamiento.

Se realizó las pruebas usando algoritmos balanceadores de carga en un Clúster de Servidores, se aplicaron en tres escenarios de 10, 30 y 50 usuario con 100, 300 y 500 peticiones respectivamente, en cuanto al tiempo de respuesta la técnica RR y HC tienen mejor desempeño y la técnica de WRR ofrece un mejor rendimiento, a mayor número de peticiones el algoritmo WRR tiene mejor desempeño tanto en tiempo de respuesta y rendimiento.

4.2. Recomendaciones.

Para futuras investigaciones hacer lo posible de aplicarlo en un entorno real, del mismo modo aplicarlos en otras arquitecturas como la nube o mediante los sistemas definidos por software.

A los administradores de servidores en lo posible implementar servidores de balanceo de carga y en la medida de ir automatizando el servidor no apresurarse a aumentar las capacidades y de esta manera poder reducir los costos.

Asimismo, tener en cuenta, que, en la elección de un algoritmo de balanceo de carga, es importante tomar en cuenta los servicios o recursos que ofrecerá el servidor.

REFERENCIAS.

[1] S. Kemp, “Digital 2021 July Global Statshot Report,” 2021. https://datareportal.com/reports/digital-2021-july-global-statshot?rq=Digital 2021 July Global Statshot Report.

[2] S. Kemp, “DIGITAL 2021: PERU,” *DataReportal*, 2021. https://datareportal.com/reports/digital-2021-peru.

[3] El Comercio, “Banco de la Nación reporta caída de sistema en plataformas digitales desde ayer,” *El Comercio*, 2019. https://elcomercio.pe/economia/peru/banco-nacion-reporta-caida-plataformas-digitales-ayer-app-banca-internet-noticia-660950-noticia/?ref=ecr.

[4] Gestión, “Usuarios reportan caída del aplicativo Interbank App desde la tarde,” *Gestión*, 2021. https://gestion.pe/economia/empresas/usuarios-reportan-caida-del-aplicativo-interbank-app-desde-la-tarde-nndc-noticia/?ref=gesr.

[5] S. Jenny, Gross; Adam, “A widespread internet outage affects major websites.,” *The New York Times*, 2021.

[6] C. M. Marcos, “Dozens of websites go down in a widespread internet outage.,” *The New York Times*, 2021. https://www.nytimes.com/2021/07/22/business/internet-outage.html?searchResultPosition=5.

[7] Downdetector, “Resumen de las interrupciones de servicio,” *Downdetector*, 2021. https://downdetector.pe/archivo/2021/10/ (accessed Aug. 13, 2023).

[8] R. Barrantes, “Redes de Internet con sobrecarga,” *PUCP*, 2020. https://puntoedu.pucp.edu.pe/voces-pucp/redes-de-internet-con-sobrecarga/ (accessed Aug. 13, 2023).

[9] I. Saraswati, S. Praptodiyono, A. S. Pramudyo, and Kurniawan, “Increasing web server performance using the web balancing method,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 673, no. 1, 2019, doi: 10.1088/1757-899X/673/1/012065.

[10] E. Rohadi, A. Amalia, A. Prasetyo, M. F. Rahmat, A. Setiawan, and I. Siradjuddin, “Cluster implementation on mini Raspberry Pi computers using Round Robin Algorithm,” *J. Phys. Conf. Ser.*, vol. 1450, no. 1, pp. 0–9, 2020, doi: 10.1088/1742-6596/1450/1/012068.

[11] H. Y. Liu, C. Y. Chiang, H. S. Cheng, and M. L. Chiang, “OpenFlow-based server cluster with dynamic load balancing,” *Proc. - 2018 IEEE/ACIS 19th Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2018*, pp. 99–104, 2018, doi: 10.1109/SNPD.2018.8441096.

[12] K. A. Jadhav, M. M. Mulla, and D. G. Narayan, “An Efficient Load Balancing Mechanism in Software Defined Networks,” *Proc. - 2020 12th Int. Conf. Comput.*

Intell. Commun. Networks, CICN 2020, pp. 116–122, 2020, doi: 10.1109/CICN49253.2020.9242601.

[13] M. L. Chiang, H. S. Cheng, H. Y. Liu, and C. Y. Chiang, “SDN-based server clusters with dynamic load balancing and performance improvement,” *Cluster Comput.*, vol. 24, no. 1, pp. 537–558, 2021, doi: 10.1007/s10586-020-03135-w.

[14] H. M. Noman and M. N. Jasim, “A Comparative Performance Analysis for Static and Dynamic Load Balancing Techniques in Software Defined Network Environment,” *J. Phys. Conf. Ser.*, vol. 1773, no. 1, pp. 0–6, 2021, doi: 10.1088/1742-6596/1773/1/012010.

[15] Y. A. H. Omer, A. B. A. Mustafa, and A. G. Abdalla, “Performance Analysis of Round Robin Load Balancing in SDN,” *Proc. 2020 Int. Conf. Comput. Control. Electr. Electron. Eng. ICCCEEE 2020*, 2021, doi: 10.1109/ICCCEEE49695.2021.9429662.

[16] J. Rathore, B. Keswani, and V. S. Rathore, *Analysis of Load Balancing Algorithms Using Cloud Analyst*, vol. 841. Springer Singapore, 2019.

[17] B. Alankar, G. Sharma, H. Kaur, R. Valverde, and V. Chang, “Experimental setup for investigating the efficient load balancing algorithms on virtual cloud,” *Sensors (Switzerland)*, vol. 20, no. 24, pp. 1–26, 2020, doi: 10.3390/s20247342.

[18] Ş. Aymaz, T. Çavdar, S. Aymaz, and E. Öztürk, “An Analysis of Load Balancing Strategies with Wireshark in Software Defined Networks,” *2018 Int. Conf. Artif. Intell. Data Process. IDAP 2018*, pp. 1–5, 2019, doi: 10.1109/IDAP.2018.8620766.

[19] M. Hamdani, Y. Aklouf, and H. Chaalal, “A Comparative Study on Load Balancing Algorithms in Cloud Environment,” *PervasiveHealth Pervasive Comput. Technol. Healthc.*, 2020, doi: 10.1145/3447568.3448466.

[20] R. Li, Y. Li, and W. Li, “An Integrated Load-balancing Scheduling Algorithm for Nginx-Based Web Application Clusters,” *J. Phys. Conf. Ser.*, vol. 1060, no. 1, 2018, doi: 10.1088/1742-6596/1060/1/012078.

[21] S. Wilson Prakash and P. Deepalakshmi, “DServ-LB: Dynamic server load balancing algorithm,” *Int. J. Commun. Syst.*, vol. 32, no. 1, pp. 1–11, 2019, doi: 10.1002/dac.3840.

[22] S. Suherman, M. Aziz, and E. B. Nababan, “Load balancing algorithm for a local video network,” *J. Phys. Conf. Ser.*, vol. 1235, no. 1, 2019, doi: 10.1088/1742-6596/1235/1/012018.

[23] A. K. Arahunashi, G. G. Vaidya, S. Neethu, and K. V. Reddy, “Implementation of Server Load Balancing Techniques Using Software-Defined Networking,” *Proc. 2018 3rd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solut. CSITSS 2018*, pp. 87–90, 2018, doi: 10.1109/CSITSS.2018.8768754.

[24] A. De León, “Servidor,” *hostingdiario*, 2019. https://hostingdiario.com/servidor/.

[25] E. A. Marchionni, *Administrador de servidores*. 2011.

[26] valentin Almirón, *Redes administración de servidores*. 2014.

[27] J. L. Villada Romero, *Instalación y configuración del software de servidor web. IFCT0509*, 1st ed. 2014.

[28] B. G. Vercher, *Administración y auditoría de los servicios web*. 2015.

[29] J. F. Martínez, *Implantación de aplicaciones web en entornos internet, intranet y extranet.* 2015.

[30] J. M. Huidobro Moya, *Sistemas telemáticos*, 3rd ed. 2005.

[31] E. A. Marchionni, *Administrador de servidores*. 2011.

[32] J. J. G. Cañizares, *Instalación y configuración del software de servidor web*, 5th ed. 2015.

[33] E. Borges, “Servidor en Cluster,” 2019. https://blog.infranetworking.com/servidor-en-cluster/.

[34] L. Fernández, “Balanceadores de Carga: Así puedes mejorar el rendimiento de tu web,” *redeszone*, 2020. .

[35] J. Guerrero Fernández, *Sistemas de almacenamiento*, 5th ed. 2015.

[36] E. Borges, “Servidor en Cluster,” *Infranetworking*, 2019. https://blog.infranetworking.com/servidor-en-cluster/.

[37] J. E. Córcoles Tendero, *Gestión de bases de datos.*, 2ª Edición. 2014.

[38] J. F. Martínez, *Implantación de aplicaciones web en entornos internet, intranet y extranet*. 2013.

[39] J. Jimenez, “Qué es Alta Disponibilidad o HA y por qué es importante en servidores,” *redeszone*, 2020. https://www.redeszone.net/tutoriales/servidores/alta-disponibilidad-importante-servidores/.

[40] A. Asimane, *Windows Server 2012 R2 - Configuración de servicios avanzados*. 2014.

[41] J. A. Caballero González, Carlos Clavero García, *Sistemas de almacenamiento*. 2016.

[42] C. L. Ainoa, *Cloud: Herramientas para Trabajar en la Nube*. 2017.

[43] H. Wacker, “¿Qué es el balanceo de carga?,” *computerworld*, 2000. https://www.computerworld.es/tendencias/que-es-el-balanceo-de-carga.

[44] J. Costas Santos, *Seguridad y Alta Disponibilidad*. 2014.

[45] CloudFlare, “What is load balancing,” *CloudFlare*. https://www.cloudflare.com/es-es/learning/performance/what-is-load-balancing/.

[46] huaweicloud, “Load Balancing Algorithms,” *huaweicloud*, 2021. https://support.huaweicloud.com/intl/en-us/usermanual-elb/elb_ug_jt_0003.html.

[47] huaweicloud, “Load Balancing Algorithms,” *huaweicloud*, 2021. https://support.huaweicloud.com/intl/en-us/productdesc-elb/elb_pro_0003.html.

[48] Redhat, “Administración del Equilibrador de carga,” *redhat*, 2014. https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/6/html-single/load_balancer_administration/index.

[49] Ibm, “Algorithms for making load-balancing decisions,” *ibm*, 2023. https://www.ibm.com/docs/en/datapower-gateways/10.0.1?topic=groups-algorithms-making-load-balancing-decisions#lbg_algorithms__wlc.

[50] huaweicloud, “Adición o eliminación de servidores backend (balanceadores de carga dedicados),” *huaweicloud*, 2020. https://support.huaweicloud.com/intl/es-us/usermanual-elb/elb_ug_hd_0003.html.

[51] G. Singh and K. Kaur, “An Improved Weighted Least Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems,” p. 2, 2018, [Online]. Available: https://www.irjet.net/archives/V5/i3/IRJET-V5I3455.pdf.

[52] N. Bonnet, *Windows Server 2012 R2:Las bases imprescindibles para administrar y configurar su servidor*. 2014.

[53] G. Lai, “Lumen Cloud Guide to HAProxy,” *Lumen*, 2015. https://es.ctl.io/knowledge-base/servers/lumen-cloud-guide-to-haproxy/.

[54] HAProxy, “The Reliable, High Performance TCP/HTTP Load Balancer,” *HAProxy*, 2020. https://www.haproxy.org/oldstuff.html.

[55] E. Borges, “¿Qué es Nginx? Características, Ventajas e Instalación,” *Infranetworking*, 2019. https://blog.infranetworking.com/que-es-nginx/.

[56] G. M. E. Baena Paz, *Metodología de la Investigación*, 1st ed. 2014.

[57] J. Cegarra Sánchez, *Metodología de la investigación científica y técnológica*. 2012.

[58] E. Silva Gómez, “Investigación Tecnológica. Concepción Metodológica en las Ciencias de la Ingeniería,” *recitiutm*, pp. 86–87, 2014, [Online]. Available: http://recitiutm.iutm.edu.ve/index.php/recitiutm/article/view/22.

[59] G. Guerrero and M. Guerrero, *Metodología de la investigación*, 2nd ed. 2014.

[60] J. A. Fernández Vásquez, C. N. Purihuamán Leonardo, O. López Regalado, and M. J. Sánchez Chero, “Metodologá de la investigación científica y tecnológica,” vol. 1, p. 9, 2021, [Online]. Available: https://colloquiumbiblioteca.com/index.php/web/article/view/95.

[61] N. Joshi and D. Gupta, *A Comparative Study on Load Balancing Algorithms in Software Defined Networking*, vol. 276. Springer International Publishing, 2019.

[62] M. Jezreel Ian C, R. B. M. Baquirin, K. S. Guevara, and D. R. Tandingan, “Fittest Job First Dynamic Round Robin (FJFDRR) scheduling algorithm using dual queue and arrival time factor: A comparison,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 482, no. 1, pp. 0–9, 2019, doi: 10.1088/1757-899X/482/1/012046.

[63] G. Liu and X. Wang, “A Modified Round-Robin Load Balancing Algorithm Based on

Content of Request," *Proc. - 2018 5th Int. Conf. Inf. Sci. Control Eng. ICISCE 2018*, pp. 66–72, 2019, doi: 10.1109/ICISCE.2018.00023.

[64] W. Chen, Z. Shang, X. Tian, and H. Li, "Dynamic Server Cluster Load Balancing in Virtualization Environment with OpenFlow," *Int. J. Distrib. Sens. Networks*, vol. 2015, 2015, doi: 10.1155/2015/531538.

[65] S. W. Prakash and P. Deepalakshmi, "Server-based Dynamic Load Balancing," *2017 Int. Conf. Networks Adv. Comput. Technol. NetACT 2017*, no. July, pp. 25–28, 2017, doi: 10.1109/NETACT.2017.8076736.

[66] A. Kumar and M. Kalra, "Load balancing in cloud data center using modified active monitoring load balancer," *Proc. - 2016 Int. Conf. Adv. Comput. Commun. Autom. ICACCA 2016*, pp. 1–5, 2016, doi: 10.1109/ICACCA.2016.7578903.

[67] H. Rai, S. K. Ojha, and A. Nazarov, "Cloud Load Balancing Algorithm," *Proc. - IEEE 2020 2nd Int. Conf. Adv. Comput. Commun. Control Networking, ICACCCN 2020*, pp. 861–865, 2020, doi: 10.1109/ICACCCN51052.2020.9362810.

[68] P. J. Kumar, N. Sivakumar, J. Prabhu, P. S. Ramesh, and P. Suganya, "Analysis of heterogeneous device characteristics in round robin based load balancing algorithm with closest data center as service broker policy in cloud," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 9, pp. 1627–1630, 2019, doi: 10.35940/ijitee.i8273.078919.

[69] K. Zatwarnicki, "Providing predictable quality of service in a cloud-based web system," *Appl. Sci.*, vol. 11, no. 7, 2021, doi: 10.3390/app11072896.

[70] A. I. El Karadawy, A. A. Mawgoud, and H. M. Rady, "An Empirical Analysis on Load Balancing and Service Broker Techniques using Cloud Analyst Simulator," *Proc. 2020 Int. Conf. Innov. Trends Commun. Comput. Eng. ITCE 2020*, pp. 27–32, 2020, doi: 10.1109/ITCE48509.2020.9047753.

[71] M. R. Mesbahi, M. Hashemi, and A. M. Rahmani, "Performance Evaluation and Analysis of Load Environments," *Conf. IEEE*, pp. 145–151, 2016.

[72] V. Velde and B. Rama, "An advanced algorithm for load balancing in cloud computing using fuzzy technique," *Proc. 2017 Int. Conf. Intell. Comput. Control Syst. ICICCS 2017*, vol. 2018-Janua, pp. 1042–1047, 2017, doi: 10.1109/ICCONS.2017.8250624.

[73] I. P. A. Suwandika, M. A. Nugroho, and M. Abdurahman, "Increasing SDN network performance using load balancing scheme on web server," *2018 6th Int. Conf. Inf. Commun. Technol. ICoICT 2018*, vol. 0, no. c, pp. 459–463, 2018, doi: 10.1109/ICoICT.2018.8528803.

[74] M. Farhoudi, P. Habibi, and M. Sabaei, "Server Load Balancing in Software-Defined Networks," *9th Int. Symp. Telecommun. With Emphas. Inf. Commun. Technol. IST 2018*, pp. 435–441, 2019, doi: 10.1109/ISTEL.2018.8661114.

[75] A. B. Prasetijo, E. D. Widianto, and E. T. Hidayatullah, "Performance comparisons of

web server load balancing algorithms on HAProxy and Heartbeat," *Proc. - 2016 3rd Int. Conf. Inf. Technol. Comput. Electr. Eng. ICITACEE 2016*, pp. 393–396, 2017, doi: 10.1109/ICITACEE.2016.7892478.

[76] A. Yu and S. Yang, "Research on web server cluster load balancing algorithm in web education system," *J. Supercomput.*, vol. 76, no. 5, pp. 3364–3373, 2020, doi: 10.1007/s11227-018-2573-5.

[77] H. Xiang, T. Zhang, and Z. Li, "Big data cloud platform server load balancing algorithm based on improved chaotic partition algorithm," *J. Phys. Conf. Ser.*, vol. 1982, no. 1, pp. 0–7, 2021, doi: 10.1088/1742-6596/1982/1/012116.

[78] L. Zhu, J. Cui, and G. Xiong, "Improved dynamic load balancing algorithm based on Least-Connection Scheduling," *Proc. 2018 IEEE 4th Inf. Technol. Mechatronics Eng. Conf. ITOEC 2018*, no. Itoec, pp. 1858–1862, 2018, doi: 10.1109/ITOEC.2018.8740642.

[79] Z. Chen, H. Zhang, J. Yan, and Y. Zhang, "Implementation and research of load balancing service on cloud computing platform in IPv6 network environment," *ACM Int. Conf. Proceeding Ser.*, pp. 220–224, 2018, doi: 10.1145/3291842.3291871.

[80] B. Aditya and T. Juhana, "A high availability (HA) MariaDB Galera Cluster across data center with optimized WRR scheduling algorithm of LVS - TUN," *Proceeding 2015 9th Int. Conf. Telecommun. Syst. Serv. Appl. TSSA 2015*, pp. 6–10, 2016, doi: 10.1109/TSSA.2015.7440452.

[81] K. Ramana and M. Ponnavaikko, "A Multi-Class Load Balancing Algorithm (MCLB) for heterogeneous web cluster," *Stud. Informatics Control*, vol. 27, no. 4, pp. 443–452, 2018, doi: 10.24846/v27i4y201808.

[82] X. Jiang, H. Yang, Y. Yang, and Z. Chen, "Cluster load balancing algorithm based on dynamic consistent hash," *J. Intell. Fuzzy Syst.*, vol. 41, no. 3, pp. 4461–4468, 2021, doi: 10.3233/JIFS-189706.

[83] H. Rai, S. K. Ojha, and A. Nazarov, "Comparison Study of Load Balancing Algorithm," in *Proceedings - IEEE 2020 2nd International Conference on Advances in Computing, Communication Control and Networking, ICACCCN 2020*, Dec. 2020, pp. 852–856, doi: 10.1109/ICACCCN51052.2020.9362905.

[84] H. Xiang, T. Zhang, and Z. Li, "Journal of Physics : Serie de conferencias Algoritmo de equilibrio de carga del servidor de plataforma en la nube de big data basado en un algoritmo mejorado de partición caótica Algoritmo de equilibrio de carga del servidor de plataforma en la nube de bi," pp. 0–7, 2021.

[85] X. Jiang, H. Yang, Y. Yang, and Z. Chen, "Cluster load balancing algorithm based on dynamic consistent hash," *J. Intell. Fuzzy Syst.*, vol. 41, no. 3, pp. 4461–4468, 2021, doi: 10.3233/JIFS-189706.

ANEXOS.

Anexo 1. Resolución de aprobación del proyecto de investigación

UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

RESOLUCIÓN N°1179--2021/FIAU-USS

Pimentel, 10 de diciembre de 2021

VISTO:
El Acta de reunión N°1611-2021 del Comité de investigación de la Escuela profesional de INGENIERÍA DE SISTEMAS remitida mediante Oficio N°0382-2021/FIAU-IS-USS de fecha 24 de noviembre de 2021, y;

CONSIDERANDO:
Que, de conformidad con la Ley Universitaria N° 30220 en su artículo 48° que a letra dice: "La investigación constituye una función esencial y obligatoria de la universidad, que la fomenta y realiza, respondiendo a través de la producción de conocimiento y desarrollo de tecnologías a las necesidades de la sociedad, con especial énfasis en la realidad nacional. Los docentes, estudiantes y graduados participan en la actividad investigadora en su propia institución o en redes de investigación nacional o internacional, creadas por las instituciones universitarias públicas o privadas.";

Que, de conformidad con el Reglamento de grados y títulos en su artículo 21° señala: "Los temas de trabajo de investigación, trabajo académico y tesis son aprobados por el Comité de Investigación y derivados a la Facultad o Escuela de Posgrado, según corresponda, para la emisión de la resolución respectiva. El periodo de vigencia de los mismos será de dos años, a partir de su aprobación. En caso un tema perdiera vigencia, el Comité de Investigación evaluará la ampliación de la misma.

Que, de conformidad con el Reglamento de grados y títulos en su artículo 24° señala: La tesis es un estudio que debe denotar rigurosidad metodológica, originalidad, relevancia social, utilidad teórica y/o práctica en el ámbito de la escuela profesional. Para el grado de doctor se requiere una tesis de máxima rigurosidad académica y de carácter original. Es individual para la obtención de un grado; es individual o en pares para obtener un título profesional. Asimismo, en su artículo 25° señala: "El tema debe responder a alguna de las líneas de investigación institucionales de la USS S.A.C.".

Que, según documentos de Vistos el Comité de investigación de la Escuela profesional de INGENIERÍA DE SISTEMAS acuerdan aprobar los temas de las Tesis a cargo de los estudiantes que se detallan en el anexo de la presente Resolución.

Estando a lo expuesto, y en uso de las atribuciones conferidas y de conformidad con las normas y reglamentos vigentes;

SE RESUELVE:

ARTÍCULO 1°: APROBAR, el tema de la Tesis perteneciente a la línea de investigación de INFRAESTRUCTURA, TECNOLOGÍA Y MEDIO AMBIENTE, a cargo de los estudiantes del Programa de estudios de INGENIERÍA DE SISTEMAS según se detalla en el anexo de la presente Resolución.

ARTÍCULO 2°: ESTABLECER, que la inscripción del Tema de la Tesis se realice a partir de emitida la presente resolución y tendrá una vigencia de dos (02) años.

ARTÍCULO 3°: DEJAR SIN EFECTO, toda Resolución emitida por la Facultad que se oponga a la presente Resolución.

REGÍSTRESE, COMUNÍQUESE Y ARCHÍVESE

Mg. Victor Alexci Tuesta Monteza
Decano (e) / Facultad De Ingeniería, Arquitectura Y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

MBA. María Noelia Slater Rivera
Secretaría Académica / Facultad de Ingeniería, Arquitectura y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

Cc: Interesado, Archivo

USS | UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

RESOLUCIÓN N°1179--2021/FIAU-USS

Pimentel, 10 de diciembre de 2021

ANEXO

N°	AUTOR(ES)	TEMA DE TESIS
1	CABRERA SANCHEZ KEVIN ALONSO MENDOZA FERRE ESPERANZA NATALY	DESARROLLO DE UNA METODOLOGÍA DE GESTIÓN DE RIESGOS PARA MEJORAR LA DISPONIBILIDAD DE SERVICIO DE TI DE UN MUNICIPIO DISTRITAL
2	ROJAS ARRUNATEGUI JOEL ENRIQUE YAFAC LAU CESAR LEONIDAS	DESARROLLO DE UN MODELO DE PROCESOS PARA LA ADQUISICIÓN DE SOFTWARE BASADO EN LA NTP-ISO/IEC 12207 PARA MEJORAR LA GESTIÓN DE LAS ADQUISICIONES DE SOFTWARE EN MICROEMPRESAS PERUANAS
3	FERNANDEZ MALUQUIS JOSE EFRAIN	ANALISIS DE ALGORITMOS BALANCEADORES DE CARGA PARA UN CLÚSTER DE SERVIDORES PARA MEJORAR LA DISPONIBILIDAD DE UN SERVIDOR
4	RAMOS SANDOVAL FABIOLA ARACELY CANTORAL MONTEJO CESAR ENRIQUE	DESARROLLO DE UN METODO DE CLASIFICACIÓN AUTOMÁTICA PARA LA DETECCIÓN EFICIENTE DEL RIESGO DE ANEMIA INFANTIL A PARTIR DE HÁBITOS DE ALIMENTACIÓN Y CUIDADOS
5	BOCANEGRA GUERRERO YERSON HUAMAN HUANCAS DERBIS	ANÁLISIS COMPARATIVO DE ARQUITECTURAS DE APRENDIZAJE PROFUNDO PARA LA CLASIFICACIÓN DE ROYA AMARILLA EN HOJAS DE CAFÉ
6	SANDOVAL CHERO CESAR ARTURO	MODELO DE LA GESTIÓN DE LA SEGURIDAD DE LA INFORMACIÓN ALINEADA A LA NORMA ISO/IEC 27001 ORIENTADO A LAS MICROEMPRESAS
7	DENNIS MAURICIO AVILES ODAR	APLICACIÓN DE BUENAS PRÁCTICAS PARA ENTORNOS DE DESARROLLO DE SOFTWARE BASADOS EN DEVOPS PARA MEJORAR LA INTEGRACIÓN Y DESPLIEGUE DE PROYECTOS EN UNA EMPRESA CONSULTORA DE LA CIUDAD DE CHICLAYO
8	RIVAS PLATA CASAS CARLOS GUALBERTO	DETECCIÓN DE CÁNCER DE PULMÓN EN IMÁGENES DE TOMOGRAFÍAS MEDIANTE PROCESAMIENTO DE IMÁGENES Y APRENDIZAJE AUTOMÁTICO
9	PECHE SANCHEZ CHRISTIAN WILFREDO	DISEÑO DE ARQUITECTURA DE MICROSERVICIOS PARA OPTIMIZAR PROCESOS EN LA GESTIÓN DE VENTAS ONLINE
10	SEVERINO HERNÁNDEZ YAMPIER GILBERTO	EVALUACIÓN DEL RENDIMIENTO DE UNA APLICACIÓN WEB CON ARQUITECTURA DE MICROSERVICIOS SOPORTADOS EN LA NUBE EN UN AMBIENTE DE ALTA CONCURRENCIA
11	CHANG HIDALGO HAWARD MIGUEL	COMPARACIÓN DE TÉCNICAS DE ESTIMACIÓN BASADAS EN MACHINE LEARNING PARA PREDECIR COSTOS EN LOS PLANES DE ADQUISICIONES DE LAS ENTIDADES PÚBLICAS DEL PERÚ
12	PUICON PISFIL MIRIAN ALICIA VILCHEZ CHANGANAQUI RICHARD ALEXIS	DESARROLLO DE UN MODELO DE PROCESOS BASADO EN ESTÁNDARES PARA LA EVALUACIÓN DE LA USABILIDAD WEB PARA MICROEMPRESAS PERUANAS
13	LOPEZ ABANTO GUILLERMO ANTONIO	EVALUACIÓN DE LA SEGURIDAD DE UN SISTEMA DE VOTACIÓN ELECTRÓNICA CON BLOCKCHAIN
14	CALDERON ZUÑIGA JESUS TELLO TANTARICO DILSON GUZMAN	DESARROLLO DE UN MODELO DE GOBERNANZA DE TI BASADO EN MARCOS DE GOBIERNO Y GESTIÓN DE TECNOLOGÍAS DE LA INFORMACIÓN PARA INSTITUCIONES PÚBLICAS PERUANAS

Anexo 2: Población de Algoritmos balanceadores de carga

Algoritmo	Identificador
Round Robin	RR
Random	R
Least Conecction	LC
Least-Loading	LL
Weighted Round Robin	WRR
Throttled	THR
Weighted ramdon selection	WRS
Weighted Least Connection	WLC
IP Hash	IPH
Load Balancer Server	SBL
Hash Consistente	HC
Dynamic Weighted Least Connection	DWLC
Dynamic Round Robin	DRR
First Come First Served	FCFS
Software Defined Network	SDN
Dynamic Load Balancer Server	DSLB

INDICE DE SIMILITUD

NOMBRE DEL TRABAJO

TESIS_FERNANDEZ_MALUQUIS_JOSE - Turnitin.docx

AUTOR

Jose Fernandez Maluquis

RECUENTO DE PALABRAS

19388 Words

RECUENTO DE CARACTERES

101613 Characters

RECUENTO DE PÁGINAS

113 Pages

TAMAÑO DEL ARCHIVO

4.7MB

FECHA DE ENTREGA

Nov 3, 2023 9:21 AM GMT-5

FECHA DEL INFORME

Nov 3, 2023 9:22 AM GMT-5

9% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base c

- 8% Base de datos de Internet
- 1% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossr
- 4% Base de datos de trabajos entregados

Excluir del Reporte de Similitud

- Material bibliográfico
- Material citado
- Coincidencia baja (menos de 8 palabras)