DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI

ISSN: 2007-6363

Publicación Semestral Pädi Vol. 13 No. 26 (2026) 149-160

Análisis del rendimiento de un servidor rocks para servicios Web

Analysis of the performance of a rocks server for Web services

C. García-Herrera[a,*], O. González-González[a], L. C. Méndez-Guevara[a]

[a] *Ingeniería en Computación, Centro Universitario UAEM Valle de Teotihuacán, Universidad Autónoma del Estado de México, Santo Domingo Aztacameca, México, México.*

Resumen

El presente documento tiene como objetivo determinar el rendimiento de un clúster homogéneo configurado como servidor Web. Se describen los pasos de instalación del sistema operativo, la verificación de los servicios implementados, la configuración de las directivas de distribución de carga y del directorio de intercambio utilizado para el envío de archivos dentro del arreglo. Asimismo, se analizan los métodos de reenvío y, finalmente, el rendimiento del sistema mediante parámetros definidos por el usuario. Para evaluar dicho rendimiento se establecen métricas como el módulo de procesamiento más eficiente, el número máximo de usuarios concurrentes, el menor tiempo de respuesta por solicitud y la configuración óptima del sistema.

Palabras Clave: Clúster, servidor, nodo, métricas, rendimiento.

Abstract

This paper aims to determine the performance of a homogeneous cluster configured as a Web server. It describes the steps for installing the operating system, verifying the services implemented, configuring load distribution policies, and setting up the swap directory used for file transfer within the cluster. In addition, forwarding methods are analyzed, and the system's performance is evaluated based on user-defined parameters. Metrics such as the most efficient processing module, the maximum number of concurrent users, the shortest response time per request, and the optimal system configuration are considered.

Keywords: Cluster, server, node, metrics, performance.

1. Introducción

El presente trabajo tiene como objetivo describir la configuración de un clúster orientado a la distribución de carga en solicitudes Web entrantes, así como la determinación de su rendimiento. El sistema implementado distribuye dichas solicitudes entre tres nodos del arreglo; cada nodo procesa una parte de las peticiones y devuelve las respuestas al nodo maestro, encargado de enviarlas finalmente al cliente. Para ello, se utiliza el sistema operativo Rocks, versión 7, cuya instalación y configuración son descritas paso a paso en este estudio.

Una vez completada la instalación en el nodo maestro, se verificó el correcto funcionamiento de los servicios Web, DHCP (Dynamic Host Configuration Protocol), SSH (Secure Shell) y Ganglia, los cuales resultan esenciales para la instalación en los nodos esclavos. Finalizada esta etapa, se procede a la administración general del clúster.

Asimismo, se introduce el sistema de monitoreo Ganglia, con el propósito de comprender las métricas empleadas por este software.

Posteriormente, se detalla el modo de operación del servidor Web, con énfasis en la función de la carpeta export, utilizada en el reenvío de datos desde el nodo maestro hacia los nodos esclavos. Se explica también la forma en que el clúster ejecuta esta tarea y se describe la distribución de carga, ya sea de manera programada o mediante la intervención de un administrador.

Seguidamente, se analizan los métodos de reenvío proporcionados por la herramienta servidor virtual de Linux y se aborda la instalación y configuración del administrador de carga conocido como servidor virtual de protocolo de Internet, empleado en la primera prueba de rendimiento.

Finalmente, se presentan las pruebas de carga controladas, configuradas por el usuario a través de parámetros como la duración de la prueba, el número de usuarios concurrentes o la emulación de un entorno de Internet. Dichas pruebas permiten determinar el rendimiento del sistema bajo diferentes condiciones de operación.

2. Marco teórico

*Autor para la correspondencia: cgarciah@uaemex.mx
Correo electrónico: cgarciah@uaemex.mx (Cozobi García-Herrera), shomarey90@gmail.com (Ossmar González González), lcmendezg@uaemex.mx (Laura Cecilia Méndez Guevara)

Historial del manuscrito: recibido el 24/10/2024, última versión-revisada recibida el 02/05/2025, aceptado el 07/10/2025, en línea (postprint) desde el 10/10/2025, publicado el 05/01/2026. **DOI:** https://doi.org/10.29057/icbi.v13i26.14025

Para comprender la construcción de un clúster es necesario definir sus características fundamentales, sus elementos constitutivos y su clasificación. Un clúster computacional puede entenderse como la interconexión de múltiples equipos mediante una red de alta velocidad, que permite sumar sus recursos individuales y conformar una supercomputadora. En términos generales, un clúster es un conjunto de computadoras que combina múltiples procesadores con el propósito de ofrecer soluciones eficientes en tiempo y costo (CIMAT, 2023).

Los principales elementos de un clúster son el hardware (componentes físicos) y el software (principalmente el sistema operativo). Este último constituye el programa esencial que permite la interacción entre la máquina y el usuario, coordinando la gestión de memoria, archivos y la comunicación entre hardware y aplicaciones (Garza, 2017).

Existen sistemas operativos orientados al uso doméstico, educativo o de oficina, así como sistemas especializados en servidores, diseñados para habilitar servicios en Internet.

El proyecto Rocks surgió con el objetivo de simplificar la implementación y administración de clústeres. Sus esfuerzos se orientaron hacia la facilidad de instalación, gestión, actualización y escalabilidad, con la finalidad de acercar el poder computacional de los clústeres a un amplio sector de la comunidad científica (Rocks, 2018).

En cuanto a la naturaleza de su construcción, los clústeres se clasifican en:

- Homogéneos: integrados por equipos con la misma arquitectura y recursos similares, lo que garantiza uniformidad en su desempeño.

- Heterogéneos: conformados por equipos que difieren en aspectos como arquitectura, sistema operativo, capacidad de procesamiento o tiempos de acceso.

El clúster desarrollado en este proyecto es de tipo homogéneo, ya que todos los nodos comparten las mismas características de hardware y utilizan un sistema operativo común.De acuerdo con su función, los clústeres pueden clasificarse en tres categorías principales (Martínez, 2009):

Clústeres de alta disponibilidad (High Availability): diseñados para garantizar la continuidad del servicio. No buscan maximizar el poder de cálculo, sino asegurar que, en caso de falla de un nodo, otros asuman sus tareas de manera rápida y transparente (González, 2016).

Clústeres de alto rendimiento (High Performance Computing): empleados en aplicaciones con cálculos intensivos, como simulaciones científicas o modelos meteorológicos. Buscan sustituir a las supercomputadoras tradicionales con alternativas más económicas (González & Rodríguez, 2008).

Clústeres de balanceo de carga (Load Balancing): orientados a dividir el trabajo entre los nodos disponibles, reduciendo errores y aumentando la capacidad de atención. Una aplicación común es el servidor Web (Martínez, 2009).

El balanceo de carga consiste en dividir un problema en un número fijo de procesos distribuidos entre las máquinas disponibles, sin considerar las diferencias de procesador o velocidad. En este contexto, el clúster propuesto en el presente estudio se clasifica como clúster homogéneo de balanceo de carga, diseñado para funcionar como servidor Web y evaluar tanto la distribución de las solicitudes entrantes como el comportamiento general del sistema.

3. Desarrollo

Una vez definidos los aspectos básicos del clúster a implementar, se describe a continuación el proceso de instalación, configuración y administración del sistema, así como el análisis de su rendimiento frente a solicitudes Web.

3.1 Introducción a Rocks Clúster

Rocks es una distribución de Linux orientada a la implementación de clústeres, inicialmente basada en RedHat y actualmente en CentOS. Desde el año 2000, el grupo Rocks ha trabajado en simplificar la instalación, administración, actualización y escalabilidad de estas infraestructuras, con el fin de hacerlas accesibles a una amplia comunidad científica (Rocks, 2018).

3.2 Requerimientos iniciales

La instalación del sistema operativo en los nodos requiere la topología mostrada en la Figura 1.

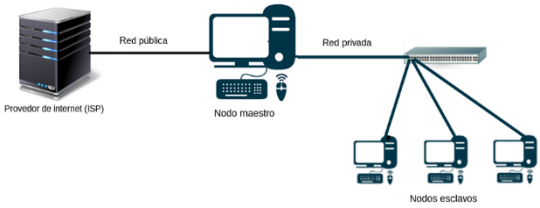


Figura 1. Conexión física del clúster.

La comunicación entre el nodo maestro y los nodos esclavos se realiza mediante una red privada, en la que cada dispositivo recibe una dirección IP asignada a través del protocolo DHCP. El nodo maestro dispone de dos interfaces de red: una dedicada a la distribución de trabajos internos y otra para la conexión pública a Internet.

En la Tabla 1 se presentan las características del hardware empleado para la construcción del clúster, compuesto por tres equipos HP con procesadores Intel Core i3 de cuatro núcleos, 4 GB de memoria RAM y discos duros de entre 120 y 320 GB.

Tabla 1. Características del hardware.

Equipos	Características
HP RP 5800	Procesador Intel Core i3 con 4 núcleos Memoria RAM 4 Gb Disco duro 320 Gb Cuenta con gabinete metálico
HP 8100 SFF	Procesador Intel Core i3 con 4 núcleos Memoria RAM 4 Gb Disco duro 320 Gb Cuenta con gabinete metálico
HP 8100 SFF	Procesador Intel Core i3 con 4 núcleos Memoria RAM 4 Gb Disco duro 120 Gb Cuenta con gabinete metálico

3.3 Ganglia (Monitor del sistema)

Antes de iniciar los trabajos en el clúster, es fundamental comprender cómo se monitorean sus recursos, ya que este proceso permite evaluar el funcionamiento general del sistema. Para este fin, se integra Ganglia dentro de la instalación, el cual incluye tres aplicaciones internas que facilitan el monitoreo (Massie, Li, Nicholes, & Vuksan, 2012).

Ganglia es un sistema distribuido, de código abierto y escalable, diseñado para entornos de cómputo de alto rendimiento. Su interfaz principal muestra gráficas generales de carga por hora para todos los nodos conectados. En la Figura 2 se observa la vista en cuadrícula de los clústeres monitoreados; en este proyecto únicamente aparece tlaloc, el único nodo activo. Las gráficas resumen la carga horaria del sistema, el volumen de datos transmitidos y recibidos, el porcentaje de uso de CPU y memoria, así como el estado de los nodos (activos o inactivos).

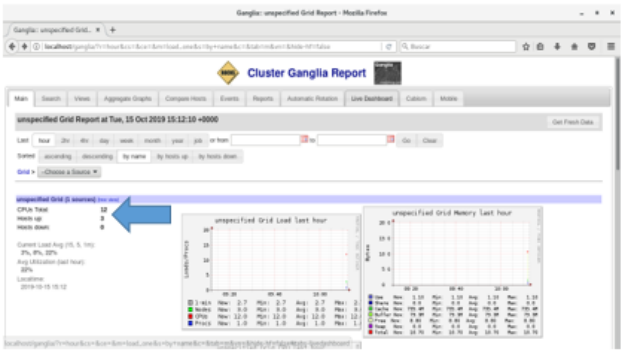


Figura 2. Portal principal de Ganglia.

3.4 El servidor Web Apache

El sistema operativo Rocks 7 incorpora el servidor Web Apache dentro de los componentes preinstalados en el clúster. Uno de los parámetros clave es el módulo de procesos múltiples (MPM), el cual puede configurarse en tres modalidades: prefork, worker y event. Estos módulos determinan la manera en que Apache procesa las solicitudes entrantes. Por defecto, el sistema utiliza prefork, con valores preestablecidos que permiten responder peticiones aun sin configuración adicional. La función esencial de los MPM consiste en aceptar las solicitudes del cliente y resolverlas mediante distintos procesos o hilos, aplicando un enfoque de tipo divide y vencerás (Apache, 2020).

El módulo Prefork implementa un servidor sin subprocesos, de modo que cada solicitud es atendida por un único proceso. Este aislamiento resulta adecuado para sitios Web que requieren evitar subprocesamiento por incompatibilidad de bibliotecas, además de que garantiza que un error en una solicitud no afecte a las demás (Apache, 2020).

El módulo Worker aplica un modelo híbrido, basado en procesos e hilos, lo que permite atender un gran número de solicitudes utilizando menos recursos que prefork. Aun así, mantiene estabilidad al conservar múltiples procesos activos para responder de forma simultánea (Apache, 2020).

El módulo Event se basa en Worker y optimiza la gestión de conexiones persistentes cliente-servidor. Una vez completada la primera solicitud, el cliente puede mantener la conexión abierta para enviar nuevas peticiones sin generar la sobrecarga de abrir procesos adicionales. Sin embargo, este esquema requiere mantener procesos e hilos en espera, incluso cuando el cliente no envía datos, lo cual representa un consumo innecesario. Para solucionarlo, event utiliza un hilo dedicado que monitorea las conexiones y las cierra automáticamente cuando quedan inactivas (Apache, 2020).

La elección del módulo depende de las exigencias del servicio que se desee ofrecer. Las principales directivas de configuración de los MPM son las siguientes (Apache, 2020):

- StartServers: número de procesos creados al iniciar el servicio.
- MinSpareServers y MaxSpareServers: exclusivas de prefork; ajustan dinámicamente la cantidad de procesos disponibles en función de la carga.
- MaxClients: número máximo de procesos o conexiones simultáneas; su valor predeterminado es 150, aunque puede configurarse hasta 256 en prefork.
- MaxRequestsPerChild: número máximo de solicitudes que atiende un proceso antes de finalizar; su valor por defecto es 4200 en prefork y 0 en worker.
- MinSpareThreads y MaxSpareThreads: exclusivas de worker, regulan la cantidad mínima y máxima de hilos disponibles.
- ThreadsPerChild: también de worker, establece el número de hilos por proceso hijo, el valor predeterminado es 25.

La Tabla 2 resume las directrices principales asociadas a cada módulo de procesamiento múltiple (prefork, worker y event).

Estos parámetros se configuran en el archivo httpd.conf, localizado en el nodo maestro, y deben replicarse en todos los nodos esclavos del clúster para garantizar un comportamiento uniforme del sistema.

Tabla 2. Directrices para cada módulo de procesamiento múltiple.

Prefork	Worker	Event
StartServers 10	StartServers 5	StartServers 3
MinSpareServers 5	MaxClients 350	MinSpareThreads 75
MaxSpareServers 25	MinSpareThreads 25	MaxSpareThreads 150
ServerLimit 256	MaxSpareThreads 75	ThreadsPerchild 75
MaxClients 256	ThreadsPerchild 25	MaxRequestPerchild 21000
MaxRequestPerchild 4200	MaxRequestPerchid 0	MaxRequestWorkers 150

La configuración mostrada en la Figura 3 corresponde al archivo httpd.conf en el nodo maestro, en el cual se empleó el módulo prefork con sus valores predeterminados. Este archivo debe replicarse en todos los nodos esclavos para mantener la coherencia en la gestión de solicitudes.

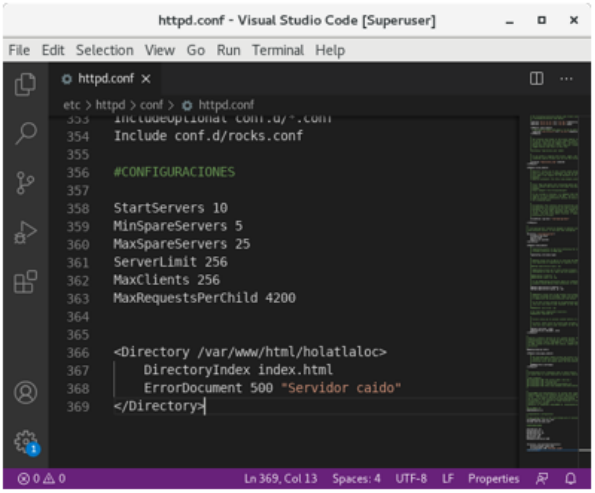


Figura 3. Configuración del archivo httpd.conf en el nodo maestro.

En cuanto al almacenamiento compartido, la carpeta /export, ubicada en el directorio raíz del nodo maestro, permite distribuir archivos hacia los nodos esclavos. Dentro de esta carpeta, el subdirectorio /export/apps contiene los elementos destinados a compartirse en el clúster. En los nodos esclavos, esta estructura se replica como /share/apps, lo que asegura la disponibilidad de los mismos archivos en todo el sistema. Como parte de la validación, se compartió la carpeta holatlaloc, utilizada en las pruebas de distribución.

3.5 Distribución de carga para peticiones Web

Una de las principales ventajas de instalar un sistema operativo orientado a clúster es la inclusión de herramientas preconfiguradas que facilitan tanto la gestión del sistema como la ejecución de tareas distribuidas entre múltiples procesadores (Ríos, 2010). En este sentido, Rocks 7 está diseñado para admitir cualquier tipo de tarea distribuida, siempre que esté programada para ejecutarse en paralelo, además de administrar y asignar de manera eficiente la carga de trabajo entre los distintos nodos.

Entre las herramientas más relevantes se encuentran LVS (Linux Virtual Server) e IPVSADM (IP Virtual Server Admin). LVS permite distribuir solicitudes de servicios como Web, FTP, DNS, VoIP o correo electrónico entre los diferentes procesadores. Esta herramienta, integrada en el kernel de Linux a partir de la versión 2.6.x, asegura una correcta distribución de carga.

Por su parte, IPVS proporciona la capa de administración necesaria para configurar los parámetros de LVS y gestionar el tráfico entrante. En la arquitectura implementada, el nodo maestro se encarga de recibir las solicitudes Web, distribuirlas a los nodos esclavos para su procesamiento y, finalmente, recopilar y enviar las respuestas de regreso a los clientes. La herramienta IPVSADM, instalada y configurada exclusivamente en el nodo maestro, cumple el papel de administrador central del flujo de solicitudes y de la interacción entre los diferentes elementos del clúster (Server, 1998).

3.6 Configuración del administrador IPVSADM

La herramienta IPVSADM permite configurar diferentes algoritmos de distribución de carga, los cuales determinan cómo se asignan las solicitudes a los nodos del clúster (RedHat, 2020a):

- rr (Round-Robin): distribuye las solicitudes de manera equitativa entre los nodos, sin considerar su capacidad o carga actual.
- wrr (Weighted Round-Robin): asigna un mayor número de solicitudes a los nodos con mayor peso configurado, mientras que los nodos con pesos menores reciben proporcionalmente menos tareas.
- lc (Least-Connection): dirige las solicitudes a los nodos con menor número de conexiones activas.
- wlc (Weighted Least-Connection, valor por defecto): distribuye las solicitudes en función de la relación entre el peso asignado a cada nodo y el número de conexiones activas.
- lblc (Locality-Based Least-Connection Scheduling): procura que las solicitudes recurrentes sean atendidas por el mismo nodo, siempre que la carga lo permita.
- dh (Destination Hash Scheduling): diseñado para servidores proxy-cache; asigna las solicitudes basándose en una tabla hash estática de direcciones IP de destino.
- sh (Source Hash Scheduling): distribuye las solicitudes de acuerdo con una tabla hash estática construida a partir de las direcciones IP de origen.

Para validar el funcionamiento del clúster y observar la aplicación de estos algoritmos, se diseñó una prueba de distribución. En ella, la carpeta holatlaloc incluyó un archivo index que identifica cada nodo esclavo por su nombre. De esta manera, al enviar solicitudes al clúster, se pudo verificar qué nodo respondía a cada petición, confirmando así la correcta distribución de la carga en función del algoritmo seleccionado.

4. **Resultados**

Para evaluar el rendimiento del servidor Web configurado en el clúster, se enviaron diferentes peticiones de manera aleatoria con el propósito de medir la capacidad de respuesta del sistema. Para ello se empleó el software Siege, una herramienta que permite configurar parámetros como el número de usuarios concurrentes, la duración de las pruebas, la simulación de un entorno en Internet y el tiempo de espera entre solicitudes.

Siege es un programa de evaluación comparativa y pruebas de carga diseñado para someter a los servidores Web a condiciones de alta demanda. Su objetivo es permitir a los desarrolladores medir el desempeño del código bajo presión y analizar la resistencia de los sistemas frente a diferentes niveles de carga (Siege Home, 2012).

La herramienta posibilita la conexión simultánea de un número configurable de clientes simulados, quienes generan solicitudes aleatorias al servidor, colocando al sistema "bajo presión" para observar su comportamiento. Siege cuenta con tres modos principales de operación:

a) Regresión
b) Simulación de Internet
c) Fuerza bruta

Para ejecutar las pruebas de manera aleatoria y simultánea, fue necesario crear el archivo urls.txt, el cual contiene las direcciones de los sitios Web disponibles en el clúster. Este archivo, ubicado en la carpeta home del equipo cliente, permitió automatizar el envío de solicitudes al arreglo, como se ilustra en la Figura 4.

Figura 4. Archivos siegerc y urls.txt.

La Figura 5 presenta la lista de carpetas y direcciones URL correspondientes a los diferentes sitios Web de prueba empleados en el proyecto. Cada sitio fue implementado en el clúster con el propósito de evaluar la distribución de carga y la capacidad de respuesta del sistema:

- camp: 192.168.0.13/camp
- barbershop: 192.168.0.13/barbershop
- viaje: 192.168.0.13/viaje
- Restaurante: 192.168.0.13/restaurante
- chapultepec: 192.168.0.13/chapu
- holatlaloc: 192.168.0.13/holatlaloc

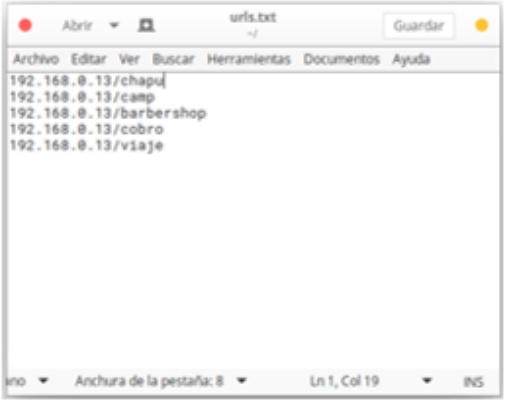
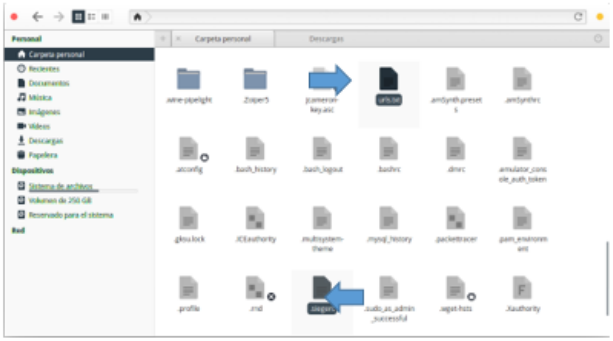


Figura 5. Configuración del archivo urls.txt

El software Siege genera, al finalizar cada prueba, un conjunto de indicadores que permiten evaluar el rendimiento del clúster (Siege Home, 2012). Estos parámetros son los siguientes:

- Transactions: número total de visitas simuladas al servidor.
- Availability: porcentaje de solicitudes atendidas correctamente por el clúster.
- Elapsed time: duración total de la prueba, expresada en segundos.
- Data transferred: volumen de datos intercambiados entre el cliente y el clúster durante la prueba.
- Response time: tiempo promedio de respuesta por solicitud.
- Transaction rate: tasa de transferencia de solicitudes por segundo.
- Throughput: cantidad promedio de bytes transferidos por segundo desde el servidor a los clientes simulados.
- Concurrency: número medio de conexiones simultáneas mantenidas durante la prueba.
- Successful transactions: total de transacciones en las que el servidor respondió con códigos inferiores a 400, incluyendo redirecciones.
- Failed transactions: número de solicitudes fallidas.
- Longest transaction: tiempo de resolución de la solicitud más lenta.
- Shortest transaction: tiempo de resolución de la solicitud más rápida.



4.1 Pruebas con Siege

El objetivo de las pruebas realizadas en el clúster fue identificar los parámetros óptimos para la distribución de solicitudes Web entrantes. Para ello se modificaron diversas variables, entre las que se destacan:

- El módulo de procesamiento de Apache.
- El número de usuarios concurrentes por prueba.
- La duración de las pruebas (para simular escenarios de carga continua).
- La simulación de entornos en Internet.
- El tiempo de espera por conexión.
- La generación de peticiones sin tiempos de retardo, con el fin de incrementar la carga sobre el sistema.

Con estas variaciones se buscó determinar:

- El número máximo de usuarios concurrentes soportados por el clúster.
- El comportamiento de los diferentes módulos de procesamiento.
- El desempeño de los distintos algoritmos de ordenamiento.
- El tiempo de respuesta mínimo alcanzado.
- Los fallos y cuellos de botella presentes en el sistema.
- El porcentaje de éxito en el procesamiento de solicitudes (con un rango esperado de entre 98 % y 100 %).
- La configuración ideal para garantizar un funcionamiento eficiente del sistema.

Las pruebas, de carácter no estandarizado, fueron diseñadas para identificar el límite de usuarios concurrentes que el clúster puede soportar antes de presentar fallos en el procesamiento de solicitudes generadas por Siege. Para ello, se modificó en cada ensayo el número máximo de clientes permitidos por nodo a través de las directivas del módulo de procesamiento en el archivo httpd.conf.

La primera prueba se llevó a cabo con la configuración predeterminada de Apache: un máximo de 150 usuarios por nodo, sin modificaciones adicionales en las directivas del módulo de procesamiento. Dado que el clúster está conformado por tres nodos, la capacidad teórica inicial fue de 450 usuarios concurrentes. En todos los casos, la distribución de carga se realizó utilizando el algoritmo de ordenamiento Round-Robin.

1. Prueba: 1 Minuto de prueba con 450 usuarios concurrentes al sitio "holatlaloc".

Los resultados de la primera prueba mostraron que el clúster soportó correctamente los 450 usuarios concurrentes previstos en la configuración predeterminada, sin registrar errores en las solicitudes procesadas. El parámetro Failed transactions se mantuvo en *0* durante toda la ejecución, confirmando la estabilidad del sistema bajo esta carga.

En esta prueba no se realizaron modificaciones en las directivas del módulo de procesamiento de Apache, ya que el objetivo fue verificar si el límite teórico de usuarios concurrentes definido por defecto podía sostenerse en la práctica sin afectar la comunicación entre el cliente y el clúster.

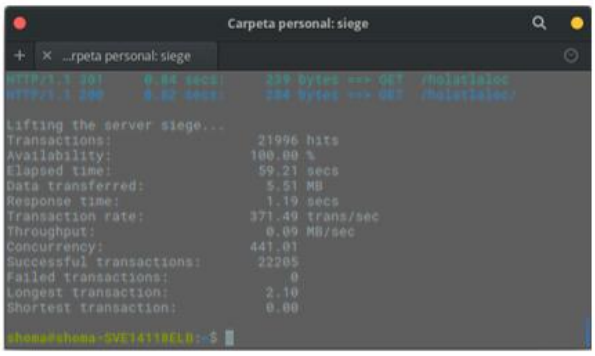


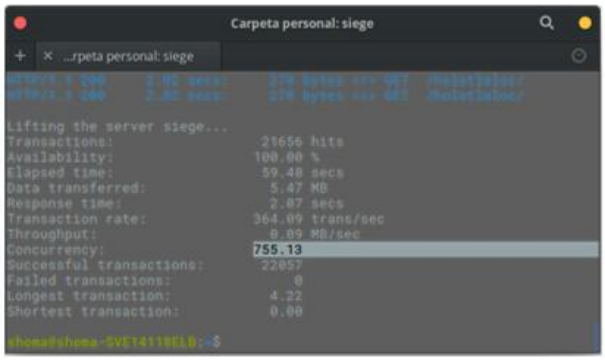
Figura 6. Resultados de la prueba 1.

2. Prueba: 1 Minuto de prueba con 768 usuarios concurrentes, módulo de procesamiento Prefork al sitio holatlaloc".

En la segunda prueba se empleó el módulo de procesamiento Prefork, configurando sus directivas conforme a la Tabla 2. La directiva MaxClients se ajustó al valor máximo permitido de 256 usuarios por nodo, lo que corresponde a una capacidad teórica de 768 usuarios concurrentes en el clúster.

Los resultados confirmaron que esta carga fue aceptada satisfactoriamente: el parámetro Failed transactions se mantuvo en *0* durante toda la ejecución, lo que indica que no se registraron errores en el procesamiento de solicitudes bajo esta configuración.

Figura 7. Resultados de la prueba 2.



3. Prueba: 1 Minuto de prueba con 1050 usuarios concurrentes al sitio "holatlaloc", con el módulo de procesamiento Worker.

En la tercera prueba se reemplazó el módulo de procesamiento Prefork por Worker, lo que permitió aumentar la directiva MaxClients a 350 usuarios por nodo. Con esta configuración, la capacidad teórica del clúster ascendió a 1050 usuarios concurrentes.

Sin embargo, al iniciar la prueba se registró una serie de errores que llevaron a su interrupción, evidenciando que el

clúster no pudo sostener tal cantidad de conexiones simultáneas. Este resultado permite delimitar la capacidad real del sistema, mostrando que el arreglo no es capaz de proporcionar servicio estable a un número tan elevado de usuarios.

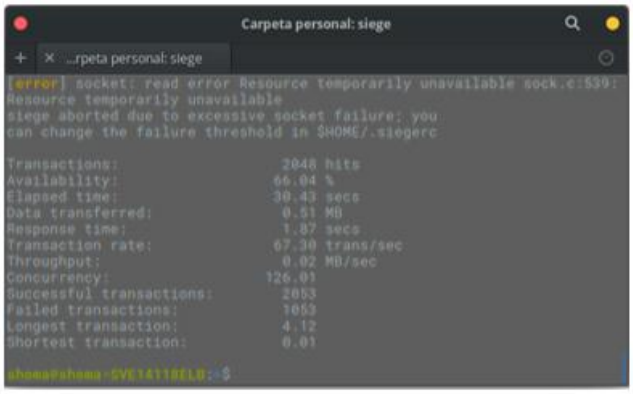


Figura 8. Resultados de la prueba 3.

3.1 Prueba: 1 Minuto de prueba con 1010, 1020 y 1030 usuarios concurrentes al sitio holatlaloc.

Con el fin de delimitar con mayor precisión la capacidad del clúster bajo el módulo Worker, se realizaron tres pruebas adicionales con valores ligeramente inferiores al límite teórico, manteniendo siempre más de 1000 usuarios concurrentes.

En particular, la prueba con 1010 usuarios concurrentes resultó satisfactoria: no se registraron errores aparentes en el envío ni en el procesamiento de solicitudes, lo que indica que esta cifra se encuentra dentro del rango de estabilidad del sistema.

Figura 9. Resultados de la prueba con 1010 usuarios.

En la prueba con 1020 usuarios concurrentes se registraron errores en el parámetro Failed transactions, aunque el proceso no se interrumpió, dado que Siege permite hasta 1024 fallos antes de abortar la ejecución (Siege Home, 2012). En total, se generaron 195 solicitudes denegadas, mientras que el resto de la prueba concluyó de manera correcta.

A partir de este resultado se determina que, con cargas iguales o superiores a 1020 usuarios concurrentes, el clúster comienza a mostrar errores en la atención de solicitudes, lo que marca un límite práctico de su capacidad bajo esta configuración.

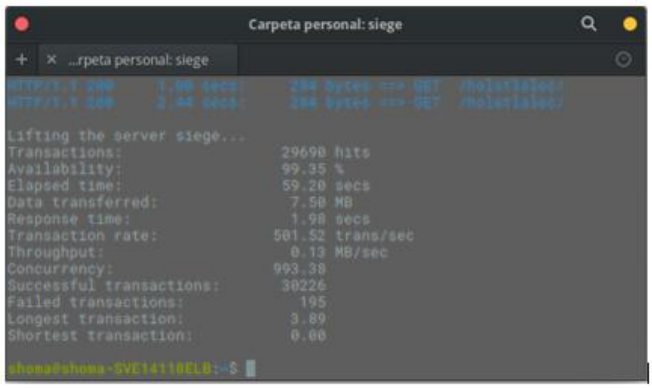
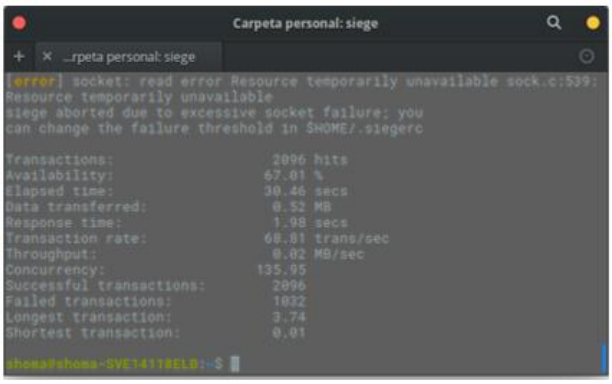


Figura 10. Resultados de la prueba con 1020 usuarios.

Como era previsible, la prueba con 1030 usuarios concurrentes abortó debido a que el clúster no logró sostener dicha carga. Esto confirma que el sistema no es capaz de operar de manera estable con un número tan elevado de conexiones simultáneas.

En consecuencia, se determina que la capacidad óptima del clúster bajo el módulo Worker se encuentra en un rango de 1010 a 1015 usuarios concurrentes, margen en el cual el sistema mantiene estabilidad y un procesamiento correcto de las solicitudes.

Figura 11. Resultados de la prueba con 1030 usuarios.



A partir de este punto, los resultados se presentan sin incluir las imágenes correspondientes con el fin de reducir la extensión del documento.

La siguiente prueba se llevó a cabo empleando el módulo de procesamiento Event. Este módulo no dispone de una directiva equivalente a MaxClients; por lo tanto, para la evaluación se adoptó como referencia el número máximo de clientes soportados previamente con el módulo Worker.

Dado que Event constituye una modificación orientada a mejorar el rendimiento de Worker, el propósito de la prueba fue analizar cuántos usuarios concurrentes podía soportar el clúster bajo esta configuración y comparar su comportamiento respecto al módulo anterior (Apache, 2020).

4. Prueba: 1 Minuto de prueba con 1010 y 1020 usuarios concurrentes al sitio holatlaloc.

La prueba con el módulo Event mostró un comportamiento similar al obtenido con Worker. En ambos casos, el clúster soportó entre 1010 y 1015 usuarios concurrentes sin presentar fallos críticos. Con 1010 usuarios, el número de errores en el parámetro Failed transactions fue menor en Event, que registró aproximadamente 70 errores menos que Worker.

En la prueba con 1020 usuarios concurrentes, el módulo Event no mostró mejoras respecto a Worker: la ejecución terminó en aborto tras superar el límite de 1024 solicitudes rechazadas. Así, puede concluirse que la capacidad práctica de ambos módulos es comparable, aunque ambos se mantienen por debajo de la capacidad teórica de 1050 usuarios. En contraste, el módulo Prefork sí logró alcanzar en la práctica el máximo calculado de 768 usuarios concurrentes.

La selección del módulo de procesamiento depende en gran medida del tipo de aplicación o sitio Web alojado en el clúster.

Prefork, módulo predeterminado de Apache, rara vez requiere modificaciones y resulta adecuado para aplicaciones Web dinámicas que utilizan bibliotecas externas, ya que ofrece mayor compatibilidad.

Worker y Event presentan limitaciones al no estar adaptados para trabajar con todas las bibliotecas externas. Este aspecto se evidenció en el presente proyecto al utilizar Ganglia, herramienta de monitoreo dinámico que emplea librerías en PHP. En este caso, los módulos Worker y Event provocaron errores en la interfaz gráfica e incluso caídas totales del sistema, mientras que Prefork aseguró un funcionamiento estable.

En consecuencia, Prefork es el único módulo plenamente adaptado para aplicaciones que dependen de bibliotecas externas, aunque los tres módulos fueron funcionales para las pruebas realizadas, dado que los sitios Web de ejemplo no requerían este tipo de compatibilidad.

Con base en los resultados anteriores, se planificaron nuevas pruebas para determinar qué módulo ofrece el mayor rendimiento y menor probabilidad de fallos. Estas evaluaciones se realizaron con la máxima cantidad de usuarios concurrentes soportada en pruebas previas, sometiendo al sistema a condiciones de carga constante. Además, se ajustó la duración de las pruebas y se habilitó la opción de simulación de entorno en Internet, con el fin de aproximarse al comportamiento real de solicitudes de usuarios.

Las pruebas se aplicaron a los cinco sitios Web de ejemplo implementados en el proyecto.

5. Prueba: 768 usuarios concurrentes, 10 minutos y módulo Prefork; 1010 usuarios concurrentes, 10 minutos y módulo Worker y 1010 usuarios concurrentes, 10 minutos y módulo Event.

En la quinta prueba se sometió al clúster a una carga constante durante un tiempo mayor que en ensayos previos, con el fin de observar la estabilidad del sistema bajo condiciones prolongadas. El objetivo fue identificar posibles errores y analizar la capacidad de cada módulo de procesamiento en estas circunstancias, observando lo siguientes:

- Prefork: alcanzó un 100 % de efectividad, procesando 77,368 solicitudes sin rechazos. El tiempo promedio de respuesta individual fue de 5.59 segundos, mientras que el tiempo total para completar la prueba fue de 599.63 segundos, con una tasa de procesamiento de 130.71 solicitudes por segundo.
- Worker: al aumentar el número de usuarios concurrentes de 768 a 1010, procesó un total de 83,637 solicitudes, es decir, 6,269 más que Prefork. No obstante, se registraron 161 solicitudes fallidas, con un porcentaje de éxito del 99.81 %.
- Event: procesó 68,295 solicitudes, cifra inferior a la alcanzada por Worker. Se contabilizaron 159 fallos, con una efectividad del 99.77 %. Su principal ventaja radicó en el tiempo de respuesta individual, aproximadamente un segundo más bajo en comparación con Prefork y Worker.

De lo anterior se puede destacar que el módulo más estable es Prefork, ya que logra niveles similares de solicitudes satisfactorias respecto a Worker y Event, pero con una menor configuración de usuarios concurrentes, lo que evita la saturación del sistema. Su principal desventaja es la imposibilidad de soportar un número mayor de usuarios en comparación con los otros módulos.

Por su parte, Worker y Event muestran mayor capacidad teórica, pero presentan más errores en pruebas prolongadas y, además, carecen de compatibilidad con bibliotecas externas. Esta limitación afecta el correcto funcionamiento de aplicaciones Web dinámicas, como Ganglia, cuya interfaz gráfica experimentó errores e incluso caídas al ejecutarse con estos módulos.

Aunque los tres módulos alcanzan niveles de efectividad entre el 99 % y 100 %, la estabilidad y compatibilidad de Prefork lo convierten en la opción más adecuada para continuar con las siguientes pruebas.

Con el módulo Prefork configurado en el clúster, se planteó incrementar el tiempo de prueba y modificar el algoritmo de distribución de carga con el objetivo de identificar cuál permite procesar un mayor número de solicitudes satisfactorias.

Dado que no todos los algoritmos de IPVSADM son compatibles con el método de reenvío NAT, se seleccionaron únicamente los siguientes (RedHat, 2020b):

- Round-Robin
- Round-Robin Weighted

- Least-Connection
- Weighted Least-Connection

Los dos últimos algoritmos ponderados (Round-Robin Weighted y Weighted Least-Connection) permiten asignar mayor carga a nodos con mayor capacidad, lo que resulta especialmente útil en clústeres heterogéneos. En este proyecto, aunque se trata de un clúster homogéneo, se configuró con mayor peso en los nodos esclavos para reducir la carga del nodo maestro, encargado de recibir y redistribuir todas las solicitudes entrantes. Las siguientes pruebas se aplicaron nuevamente a los cinco sitios Web de ejemplo implementados en el proyecto.

6. Prueba: 768 usuarios concurrentes, 30 minutos de prueba, modelos de procesamiento Prefork y módulo de ordenamiento Round-Robin y 768 usuarios concurrentes, 30 minutos de prueba, modelos de procesamiento Prefork y módulo de ordenamiento Least-Connection.

La prueba con el algoritmo de ordenamiento Round-Robin mostró un aumento en el número de solicitudes procesadas correctamente, resultado atribuido al mayor tiempo de ejecución en comparación con pruebas previas. Se registraron únicamente 7 errores, con una tasa de 125.42 transacciones por segundo. Sin embargo, el resultado más relevante fue el tiempo de respuesta promedio, que alcanzó 6.01 segundos, valor considerablemente alto para los estándares de servicio Web.

Con el algoritmo Least-Connection, se obtuvo una ligera reducción en el tiempo de respuesta —en el orden de milisegundos—, aunque no lo suficiente para representar una mejora significativa. En esta prueba se contabilizaron 223,352 solicitudes satisfactorias, es decir, 3,570 más que con Round-Robin, junto con 10 solicitudes fallidas.

Ambos algoritmos incrementaron el número total de solicitudes procesadas debido a la mayor duración de la prueba, lo que permitió observar el comportamiento del sistema bajo cargas más elevadas. El algoritmo Least-Connection se mostró más eficiente en la distribución de solicitudes, aunque los tiempos de respuesta continuaron siendo elevados.

Este aspecto resulta crítico, ya que el tiempo de respuesta recomendado para servidores Web es de 200 milisegundos a fin de garantizar un posicionamiento adecuado en motores de búsqueda (Google, 2020).

En contraste, los resultados obtenidos en estas pruebas (valores superiores a 6 segundos) son considerablemente mayores, lo que revela la existencia de cuellos de botella en el sistema.

Las pruebas siguientes se enfocarán en reducir el tiempo de respuesta mediante la modificación del número de usuarios concurrentes, la elección del algoritmo de distribución más eficiente y la optimización de los sitios Web de prueba.

6.2 Prueba: 768 usuarios concurrentes, 30 minutos de prueba, modelos de procesamiento Prefork módulo de ordenamiento Round-Robin-Weighted y módulo de ordenamiento Weigthed-Least-Connection, ambos con una petición para el nodo maestro por cada cinco a los nodos esclavos.

En la prueba con el algoritmo Round-Robin Weighted, el nodo maestro recibió una solicitud por cada cinco enviadas a los nodos esclavos, lo que redujo parcialmente la carga sobre él. Sin embargo, no se obtuvo una mejora significativa en el tiempo de respuesta, que permaneció por encima de los 6 segundos. El número de solicitudes satisfactorias fue de 221,894, valor similar al registrado en la prueba con Round-Robin.

La prueba con Weighted Least-Connection proporcionó resultados cercanos a los obtenidos con Least-Connection, aunque en este caso el nodo maestro recibió una menor proporción de solicitudes. A pesar de ello, el tiempo de respuesta no mejoró debido al elevado número de usuarios concurrentes, lo cual generó retardos en el sistema. En total, se registraron 223,202 solicitudes satisfactorias y 0 errores (Nuñez, 2014).

Los resultados evidencian que en las pruebas iniciales los tiempos de respuesta eran menores porque se utilizaba la carpeta *holatlaloc*, que no contenía un sitio Web completo. Al aumentar la complejidad de los sitios de prueba (imágenes, videos, sonido, etc.), se incrementa la latencia, definida como el tiempo que tarda en transmitirse un paquete dentro de la red (INC, 2020).

En escenarios de alta concurrencia, la transferencia de múltiples recursos Web por conexiones simultáneas provoca una saturación en el canal de comunicación entre el cliente y el clúster, lo que incrementa los tiempos de respuesta.

Para evidenciar el cuello de botella, se realizó una prueba con 768 usuarios concurrentes durante la descarga de recursos Web. En este escenario, el nodo maestro distribuye las solicitudes entre los nodos esclavos, los cuales procesan la información y devuelven las respuestas al maestro. Posteriormente, este envía al cliente los archivos que componen los sitios Web.

La limitación principal se encuentra en el ancho de banda del nodo maestro, cuya tarjeta de red alcanza un máximo de 100 Mbps. Esta capacidad resultó insuficiente para atender simultáneamente todas las solicitudes, generando retrasos en la salida de información y confirmando la existencia de un cuello de botella en la comunicación.

Una estrategia para mejorar el tiempo de respuesta del clúster consiste en reducir el número de usuarios concurrentes y, en consecuencia, disminuir el volumen de datos transmitidos en el canal de comunicación. Esta medida mitiga la saturación y evita que los cuellos de botella limiten el rendimiento y la escalabilidad del sistema (Oracle, 2009).

Con este propósito, se diseñó una nueva prueba con 384 usuarios concurrentes durante 10 minutos en los cinco sitios Web de ejemplo, con el fin de analizar si la reducción de carga se traduce en una mejora significativa en los tiempos de respuesta.

7. Prueba: 384 usuarios concurrentes, 10 minutos de prueba, modelos de procesamiento Prefork módulo de ordenamiento Round-Robin.

La prueba 7 se llevó a cabo con la mitad de los usuarios concurrentes soportados por el clúster. Los resultados mostraron una reducción significativa en el tiempo promedio de respuesta individual, que pasó a 3.74 segundos. En comparación, la prueba 5, realizada con parámetros similares, pero con mayor concurrencia, registró un tiempo de 5.59 segundos por solicitud. Esto representa una mejora de 1.85 segundos en el tiempo de entrega de respuestas.

Con base en estos hallazgos, se planteó una prueba adicional utilizando 384 usuarios concurrentes durante un periodo de 30 minutos, con el fin de evaluar la estabilidad del sistema bajo una carga intermedia sostenida en el tiempo.

7.1 Prueba 384 usuarios concurrentes, 30 minutos de prueba, modelos de procesamiento Prefork y módulo de ordenamiento Round-Robin.

En la prueba 7.1 se incrementó el tiempo de duración del ensayo con el objetivo de analizar si el tiempo de respuesta se mantenía por debajo de los 3 segundos. El propósito fue confirmar que la reducción de usuarios concurrentes, aun con cargas prolongadas, contribuye a minimizar el cuello de botella y mantiene la estabilidad del sistema. Los resultados evidencian que el clúster es capaz de soportar la cantidad máxima de usuarios concurrentes definida para esta prueba sin comprometer el procesamiento, aunque se registró un incremento en la carga de la red asociado a dicho nivel de concurrencia.

Adicionalmente, se evaluó el efecto de la optimización del contenido de los sitios Web en el tiempo de respuesta. En este proyecto, tres de los sitios de ejemplo incluyen recursos como imágenes y videos, cuyo peso puede incrementar la latencia. La minimización de estos archivos constituye una estrategia para reducir el tiempo de descarga por parte del cliente (Axarnet, 2020).

Como parte del experimento, se planteó eliminar el video de 24 MB incluido en el sitio Barbershop, con el fin de observar si esta optimización se reflejaba en una mejora en los tiempos de respuesta. La prueba se aplicó a los cinco sitios Web de ejemplo implementados en el proyecto.

7.2 Prueba: 384 usuarios concurrentes, 10 minutos de prueba, modelos de procesamiento Prefork módulo de ordenamiento Round-Robin.

La prueba 7.2 mostró una disminución de un segundo en el tiempo de respuesta respecto a la prueba 7.1. Sin embargo, esta reducción no es suficiente para concluir que el video del sitio Barbershop fuera el único factor responsable del retardo observado.

En este proyecto se integraron tres sitios Web de ejemplo con contenido multimedia (imágenes y videos), lo que permite simular un entorno más cercano al comportamiento real de un sitio Web moderno y medir el impacto de estos recursos en la latencia. Como punto de comparación, se empleó el sitio viaje, cuya estructura es básica y compuesta únicamente por texto, con el fin de observar el tiempo de respuesta que ofrece el clúster en condiciones mínimas de carga de recursos.

7.3 Prueba: 384 usuarios concurrentes, 10 minutos de prueba, modelos de procesamiento Prefork, módulo de ordenamiento Round-Robin, y prueba al sitio "viaje".

La prueba 7.3 mostró una reducción significativa en el tiempo de respuesta, alcanzando un promedio de 0.97 segundos por solicitud individual (Google, 2020). Este resultado confirma que el peso de los archivos multimedia en los sitios Web de ejemplo constituye un factor determinante en los retardos observados. En efecto, los sitios que incluyen videos e imágenes presentan tiempos de resolución más altos que aquellos compuestos únicamente por texto.

El diseño experimental contempló el uso de sitios enriquecidos con contenido multimedia para evaluar el rendimiento real del clúster, considerando tanto sus capacidades como sus limitaciones. A fin de mejorar los tiempos de respuesta, se planteó reducir el peso de los recursos multimedia en los cinco sitios Web. Para ello, se emplearon versiones optimizadas de cada sitio, reemplazadas dentro del clúster mediante la carpeta compartida /export.

La siguiente prueba (7.4) se llevó a cabo sobre estos cinco sitios Web en su versión optimizada, con el propósito de analizar la mejora alcanzada en los tiempos de respuesta.

7.4 Prueba: 384 usuarios concurrentes, 10 minutos de prueba, modelos de procesamiento Prefork y el módulo de ordenamiento Round-Robín.

La prueba evidenció una mejora en el tiempo de respuesta promedio, que se redujo a 2.10 segundos, frente a los 2.95 segundos registrados en la prueba 7.2. Esta disminución permitió incrementar la cantidad de transacciones entre el clúster y los clientes, lo que se tradujo en un mayor número de solicitudes atendidas satisfactoriamente dentro del mismo intervalo de prueba.

La optimización de los cinco sitios Web generó un aumento de 32,914 solicitudes satisfactorias adicionales respecto a la prueba 7.2, bajo condiciones similares de tiempo de prueba y usuarios concurrentes. Asimismo, se observó que al disminuir el número de usuarios concurrentes es posible obtener tiempos de respuesta aún más reducidos.

Para confirmar lo antes dicho, se planteó una nueva prueba con 200 usuarios concurrentes, cuyo propósito fue analizar el impacto directo de la concurrencia reducida sobre los tiempos de respuesta del clúster.

7.5 Prueba: 200 usuarios concurrentes, 10 minutos de prueba, modelos de procesamiento Prefork y el módulo de ordenamiento Round-Robin.

Los resultados mostraron una reducción adicional de un segundo en el tiempo de respuesta respecto a la prueba 7.4. Esto confirma que, al disminuir el número de usuarios concurrentes, el canal de comunicación presenta menor saturación, lo que facilita

que un mayor número de solicitudes se resuelva de manera correcta. Sin embargo, esta mejora implica un sacrificio en la capacidad máxima de usuarios concurrentes que el sistema puede atender de forma simultánea.

Con base en estos resultados, se planteó analizar el desempeño de los algoritmos de ordenamiento restantes bajo condiciones de menor concurrencia. Para ello, se configuraron nuevas pruebas con 200 usuarios concurrentes, con el objetivo de mantener los tiempos de respuesta por debajo de los 2 segundos y evaluar si este parámetro puede reducirse aún más mediante la elección de un algoritmo de distribución más eficiente.

7.6 Prueba: 200 usuarios concurrentes, 10 minutos de prueba, modelos de procesamiento Prefork con el módulo de ordenamiento Round-Robin-Weighted.

La prueba mostró resultados muy similares a los de la prueba 7.5, a pesar de que el algoritmo Round-Robin Weighted distribuye la carga asignando una menor proporción de solicitudes al nodo maestro (una por cada cinco entregadas a los nodos esclavos).

En este escenario, el tiempo de respuesta se redujo apenas 1 milisegundo, mientras que el número de solicitudes satisfactorias aumentó en 1,000 transacciones adicionales respecto a la prueba anterior.

7.7 Prueba: 200 usuarios concurrentes, 10 minutos de prueba, modelos de procesamiento Prefork y el módulo de ordenamiento Least-Connection.

La prueba 7.7, realizada con el algoritmo de ordenamiento Least-Connection, mostró un tiempo de respuesta prácticamente equivalente al de las pruebas 7.5 y 7.6, con una mejora marginal de 3 milisegundos. La principal diferencia radicó en el mayor número de transacciones enviadas, lo que permitió procesar y resolver un volumen superior de solicitudes de manera satisfactoria. Cabe destacar que durante esta prueba no se registraron errores.

7.8 Prueba: 200 usuarios concurrentes, 10 minutos de prueba, modelos de procesamiento Prefork y el módulo de ordenamiento Weigthed-Least-Connection.

Esta última prueba, realizada con el algoritmo de ordenamiento Weighted Least-Connection, proporcionó resultados muy similares a los obtenidos en la prueba 7.6. En ambos casos se asignó el mismo valor de importancia al nodo maestro, el cual recibió una solicitud por cada cinco asignadas a los nodos esclavos.

El análisis de las pruebas 7.5 a 7.8 mostró valores de rendimiento similares en cuanto a tiempos de respuesta y solicitudes satisfactorias. Sin embargo, el algoritmo Least-Connection se destacó por procesar una mayor cantidad de transacciones y reducir ligeramente el tiempo de respuesta por solicitud individual. En consecuencia, este algoritmo fue seleccionado para configurarse en el proyecto como opción preferente de distribución de carga.

La evaluación integral de todas las pruebas permitió establecer los parámetros de operación más adecuados para el clúster. Se determinó que la capacidad máxima práctica es de 768 usuarios concurrentes, aunque este nivel provoca un cuello de botella en el canal de comunicación, lo que retrasa la entrega de respuestas al cliente. Para mitigar esta limitación, se optó por reducir el número de usuarios concurrentes, lo que permitió mantener los tiempos de respuesta por debajo de los 2 segundos.

Asimismo, la optimización del peso de los archivos multimedia que conforman los sitios Web de prueba mejoró notablemente el rendimiento, alcanzando un tiempo mínimo de 1.09 segundos por respuesta individual, el mejor registrado en el proyecto.

En cuanto a los módulos de procesamiento de Apache, se concluyó que Prefork es el más adecuado para este proyecto. Aunque Worker y Event admiten un mayor número de usuarios concurrentes, presentan desventajas críticas:

- Incompatibilidad con bibliotecas externas, lo que afecta aplicaciones Web dependientes de estas (como Ganglia).
- Mayor probabilidad de generar cuellos de botella al elevar la concurrencia permitida.

Por el contrario, Prefork ofrece plena compatibilidad con bibliotecas externas y asegura un funcionamiento estable, aun cuando el número máximo de usuarios concurrentes es más reducido (Apache, 2020).

5. Conclusiones

El presente trabajo permitió la construcción y configuración de un clúster de tres nodos empleando el sistema operativo Rocks. Para su implementación se utilizaron protocolos y herramientas clave como DHCP para el direccionamiento, SSH para las conexiones seguras y Ganglia para el monitoreo de recursos.

Las pruebas realizadas permitieron analizar el desempeño del clúster bajo distintos escenarios, identificando los parámetros que influyen en su rendimiento:

Capacidad de usuarios concurrentes:

- El clúster alcanzó un máximo teórico de 768 usuarios concurrentes con Prefork, sin presentar errores.
- Los módulos Worker y Event permitieron configurar hasta 1050 usuarios teóricos, aunque en la práctica la capacidad se limitó a 1010–1020 usuarios, presentando fallos cuando se excedió esta cifra.

Estabilidad de los módulos de procesamiento:

- Prefork resultó el módulo más estable, con un rendimiento sostenido incluso en pruebas prolongadas de hasta 30 minutos.
- Worker y Event mostraron mayor concurrencia, pero errores en pruebas extendidas y limitaciones de compatibilidad con bibliotecas externas como Ganglia.

Algoritmos de ordenamiento (IPVSADM):

- Se evaluaron los algoritmos Round-Robin, Round-Robin Weighted, Least-Connection y Weighted Least-Connection.
- El algoritmo Least-Connection destacó por ofrecer la mayor cantidad de solicitudes satisfactorias y un tiempo de respuesta ligeramente menor, por lo que fue seleccionado como el más eficiente para este proyecto.

Cuello de botella y tiempo de respuesta:

- Se identificó un cuello de botella en el canal de comunicación del nodo maestro, vinculado a la saturación de la velocidad del canal (100 Mbps).
- La reducción del número de usuarios concurrentes y la optimización del contenido multimedia de los sitios Web permitieron mejorar el tiempo de respuesta hasta 1.09 segundos por solicitud, el mejor resultado del estudio.

En síntesis, el clúster configurado con Prefork como módulo de procesamiento y el algoritmo Least-Connection como esquema de distribución ofrece la mejor relación entre estabilidad, compatibilidad y eficiencia. Si bien la concurrencia máxima práctica se sitúa en 768 usuarios, las estrategias de optimización aplicadas demostraron que es posible mantener tiempos de respuesta competitivos y mitigar los efectos de los cuellos de botella, asegurando un rendimiento confiable para aplicaciones Web.

Referencias

Apache. (2020). Apache HTTP Server Project. Apache Software Foundation. https://httpd.apache.org/

Axarnet. (2020, 24 de febrero). Reducir el tiempo de respuesta del servidor. Axarnet. https://axarnet.es/blog/reducir-tiempo-respuesta-servidor?dt=1614621127478

CIMAT. (2023). Proyecto Clúster El Insurgente. Centro de Investigación en Matemáticas. https://hpc.cimat.mx/proyectos/Insurgente

Garza, L. (2017). Proyecto de servidor Web en clúster con alta disponibilidad y distribución de carga: Herramienta de virtualización KVM [Tesis de maestría, Universidad Politécnica de Valencia].

González, B. (2016). Clúster de alta disponibilidad sobre plataformas GNU/Linux (VERITAS) [Tesis de maestría, Universitat Oberta de Catalunya].

González, R., & Rodríguez, S. (2008). Diseño e implementación de un clúster tipo Beowulf para el desarrollo de cómputo científico avanzado. Instituto Politécnico Nacional.

Google. (2020, 17 de diciembre). PageSpeed Tools. https://developers.google.com/speed/docs/insights/Server?hl=es-419

Inc. (2020, 15 de febrero). Qué es la latencia y cómo se puede mejorar. Inc Hosting. https://www.inchosting.pe/blog/sitio-Web/que-es-la-latencia-y-como-se-puede-mejorar

Massie, M., Li, B., Nicholes, B., & Vuksan, V. (2012). Monitoring with Ganglia. O'Reilly Media.

Martínez, F. (2009). Creación y validación de un clúster de computación científica basado en Rocks [Tesis de maestría, Universidad Carlos III de Madrid].

Mesa, M. (2008). Método para el manejo del balanceo de carga en sistemas de cómputo distribuido de alto desempeño [Tesis de maestría, Universidad Nacional de Colombia].

Nuñez, J. L. (2014). Alternativas para la escalabilidad de aplicaciones en plataformas Web de alta concurrencia. Interfases, (7), 45–58.

Oracle. (2009). Identificación rápida de cuellos de botella: Una mejor manera de realizar pruebas de carga. Oracle Corporation.

RedHat. (2020a, 5 de julio). Sinopsis de la suite para clúster. Red Hat. https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/5/html-single/cluster_suite_overview/index

RedHat. (2020b, 25 de julio). Virtual Server Administration. Red Hat. https://access.redhat.com/documentation/es-es/red_hat_enterprise_linux/5/html-single/virtual_server_administration/index

Ríos, I. V. (2010). Instalación y configuración de un clúster de alto rendimiento [Tesis de maestría, Universidad Carlos III de Madrid].

Rocksclusters. (2018, 13 de mayo). Rocksclusters website. http://www.rocksclusters.org

Server, L. V. (1998, 10 de julio). Servidor virtual a través de NAT. Linux Virtual Server. http://www.linuxvirtualserver.org/VS-NAT.html

Siege Home. (2012). Joe Dog Software: Siege. https://www.joedog.org/siege-home/