Universidad de las Ciencias Informáticas

Facultad 1

Centro de Software Libre


# Solución informática para la selección de Apache 2 y Nginx durante la migración a código abierto

Trabajo final presentado en opción al título de

Máster en Informática Avanzada

**Autora:**

Ing. Nurisel Palma Pérez

**Tutores**:

Dr. Arturo Orellana García

Msc. Yoandy Pérez Villazón

**La Habana, enero de 2019**

## DEDICATORIA

*A mis padres por confiar en mí e impulsarme a seguir adelante.*

## AGRADECIMIENTOS

*A mis padres y mi hermano por ser mi sostén y mi fortaleza, la confianza que me permite proponerme otro reto y saber que siempre estarán a mi lado dándome las fuerzas necesarias.*

*A mi esposo por sus sabios consejos, por su amor incondicional, por estar a mi lado en todo momento y compartir su vida conmigo.*

*A mi sobrino Rodrigo por alegrarme cada día con su sonrisa.*

*A toda mi familia y la de mi esposo, por apoyarme y estar siempre ahí para mí.*

*Al Centro de Reproducción Asistida de Güines, el mejor equipo médico que he conocido, por mejorar mi calidad de vida y darme el ánimo para culminar este sueño.*

*A Yadiel y Yasiel, mis grandes amigos y más que amigos hermanos, por quererme y cuidarme tanto, por alentarme a vivir juntos esta experiencia, por formar el equipo perfecto "Palma Pérez Villazón", por siempre estar juntos en las buenas y las malas.*

*A mis eternas amigas Dayli y Yucel, por su amistad que rompe barreras y no importa el tiempo o la distancia, siempre estamos unidas.*

*A mis amigos María Leisy, Yosel, Ivaniet y Gustavo por su apoyo para el desarrollo de la presente tesis.*

*A todos los que formaron parte del equipo de desarrollo de HMAST.*

*A todos los integrantes de CESOL y especialmente del departamento SIMAYS.*

*A mis compañeros de guagua por preocuparse por mí.*

*A mis compañeros del grupo especial de la maestría por las inolvidables experiencias vividas.*

*A la profesora Roxana y demás profesores de los cursos, así como los tribunales de los seminarios de investigación, por el tiempo dedicado y las aceptadas recomendaciones.*

*A mis tutores y el presente tribunal por contribuir en mi formación profesional.*

*A la universidad por permitirnos la superación y ser mejores profesionales cada día.*

*A todos, gracias.*

## DECLARACIÓN JURADA DE AUTORÍA

Declaro por este medio que yo Nurisel Palma Pérez, con carné de identidad 90050624830, soy la autora principal del trabajo final de maestría “Solución informática para la selección de Apache 2 y Nginx durante la migración a código abierto”, desarrollada como parte de la Maestría en Informática Avanzada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los ___ días del mes de _____ del año 2019.

_____

Nurisel Palma Pérez

## RESUMEN

La presente investigación se centró en el objetivo de desarrollar una solución informática para aumentar la eficiencia en la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto. Se aplicó el método Analítico-Sintético para el estudio de Apache 2 y Nginx y la determinación de las particularidades de cada uno. La realización de un estudio de caso aplicando la Norma Cubana ISO/IEC 25023:2017 y el método estadístico, permitió el procesamiento de la información recopilada acerca del comportamiento de Apache 2 y Nginx, esto sirvió de base para formular la teoría relacionada con la eficiencia de ambos servidores. A partir del tipo de contenido y del total de peticiones concurrentes se definieron 60 escenarios para la selección del servidor más eficiente, teniendo en cuenta todos los indicadores o cada medida específica: Rendimiento, Utilización de recursos y Capacidad. Las tres variantes a seleccionar son: Apache 2, Nginx o Nginx como proxy inverso de Apache 2. El resultado obtenido se materializa en la Herramienta para la Migración y Administración de Servicios Telemáticos, la cual posee un componente Web que permite la selección del servidor web que más se ajusta a la institución y la activación del módulo correspondiente para su administración. La solución se validó a través de un estudio de caso, del criterio de expertos en su variante Delphi y la técnica de Iadov. Finalmente la triangulación metodológica permitió confirmar el resultado satisfactorio de todos los métodos aplicados y el cumplimiento del objetivo planteado.


**Palabras clave:** Apache 2, eficiencia, Nginx, solución informática.

## ABSTRACT

The present research focused on the objective of developing a computer solution to increase efficiency in the selection of Apache 2 and Nginx web servers during the open source migration process. The Analytical-Synthetic method was applied for the study of Apache 2 and Nginx and the determination of the particularities of each one. The realization of a case study applying the Cuban Standard ISO/IEC 25023:2017 and the statistical method, allowed the processing of the information collected about the behavior of Apache 2 and Nginx, this served as the basis to formulate the theory related to the efficiency of both servers. From the content type and the total of concurrent request, 60 scenarios were defined for the most efficient server selection, taking into account all the indicators or each specific measure: Performance, Resource utilization and Capacity. The three variants to select are: Apache 2, Nginx or Nginx as a reverse proxy of Apache 2. The result obtained is materialized in the Tool for Migration and Administration of Telematic Services, which has a Web component that allows the selection of the web server that more it adjusts to the institution and the activation of the corresponding module for its administration. The solution was validated through a case study, expert criteria in its Delphi variant and the Iadov technique. Finally, the methodological triangulation allowed to confirm the satisfactory result of all the applied methods and the fulfillment of the proposed objective.


**Keywords:** Apache 2, efficiency, Nginx, computer solution.

# ÍNDICE

# INTRODUCCIÓN

Con el transcurso de los años el número total de usuarios en Internet a nivel mundial ha aumentado significativamente. El acceso más fácil a las computadoras, la modernización de los países de todo el mundo y una mayor utilización de los teléfonos inteligentes ha dado a las personas la oportunidad de utilizar Internet con más frecuencia y más comodidad. La penetración de Internet a menudo se relaciona con el estado actual de desarrollo de las redes de comunicaciones. Hasta marzo de 2017, habían aproximadamente 731 millones de usuarios de Internet en China y 287 millones de usuarios de Internet en los Estados Unidos (STATISTA, 2017a). Las estadísticas ofrecidas por la compañía Statista muestran el número de usuarios de Internet en todo el mundo de 2009 a 2018, en América Latina y el Caribe la cifra ascendió de 186,9 millones en 2009 a 438,25 millones en 2018 (STATISTA, 2018b). Internet también ha penetrado en los diez países más poblados de Latinoamérica, siendo Cuba uno de ellos con 4,4 millones de usuarios (STATISTA, 2018c).

Internet ha creado un nuevo escenario en el que las relaciones personales cobran protagonismo. Las nuevas plataformas y herramientas colaborativas han producido un cambio desde una web 1.0 basada en páginas estáticas, meramente informativas, sin capacidad de generar una participación del usuario, hacia una web dinámica donde se produce una interrelación que genera una suma de conocimientos y/o experiencias. La web 2.0 o web social son personas colaborando, compartiendo y participando en un canal multidireccional abierto que permite lograr la máxima interacción entre los usuarios y les ofrece nuevas posibilidades de colaboración, expresión y participación (SCHIAVI, 2013). Las redes sociales son una de las actividades en línea más populares, conectarse con familiares y amigos, expresar opiniones, entretenimiento y compras en línea, se encuentran entre las razones más populares para el uso de Internet (STATISTA, 2017a).

Los servidores web desempeñan un papel dominante en la infraestructura de estos servicios. Su tarea principal es recibir y procesar solicitudes de clientes que exigen objetos específicos de los servidores y devolverlos mediante respuestas relacionadas. Los mensajes de solicitud y respuesta asociados se transportan mediante el Protocolo de Transferencia de Hipertexto (HTTP, del inglés *Hypertext Transfer Protocol*) o el Protocolo de Transferencia de Hipertexto Seguro (HTTPS, del inglés *Hypertext Transfer Protocol Secure*) mediante conexiones TCP (del inglés *Transmission Control Protocol*) entre los clientes y el servidor. El rendimiento resultante de la prestación de servicios depende crucialmente del procesamiento adecuado y eficiente de estas tareas (VAN, KRIEGER y CHAKKA, 2008). Existen varios tipos de servidores web en cuanto a la forma como procesan las peticiones que reciben de los usuarios: basados en procesos, basados en hilos y basados en eventos.

Actualmente son numerosos los servidores web, dentro de los que se encuentran Microsoft-IIS, Apache 2, Nginx, LiteSpeed, Google, Sun Java System, Lighttpd, IBM Servers y Yahoo Traffic Server. Algunos son más usados que otros debido a que poseen características distintivas que marcan la diferencia en cuanto a su selección. La información sobre el uso de servidores web es proporcionada por algunos sitios especializados en ofrecer estadísticas de diferentes tecnologías en la web. Ejemplo de ello es el sitio de la compañía inglesa Netcraft que realiza mensualmente mediciones de uso (PALMA, 2013). Según el reporte del mes de noviembre de 2018, los cuatro principales proveedores más importantes son Microsoft (39,47%), Nginx (21,71%), Apache (20,68%) y Google (1,46%) (NETCRAFT, 2018).

Por otra parte, Web Technology Surveys (W3Techs) también proporciona estadísticas del porcentaje de sitios web que hacen uso de determinado servidor web. En el reporte de noviembre de 2018 las estadísticas fueron: Apache (47,1%), Nginx (37,5%), Microsoft-IIS (10,1%), LiteSpeed (3,2%) y Google (1.0%) (W3TECHS, 2018). A partir de las estadísticas en ambos reportes, se calculó la media del porcentaje de uso de cada uno de los cuatro servidores web que coinciden, determinándose Apache con 33,89%, Nginx con 29,61%, Microsoft-IIS con 24,79% y Google Servers con 1,23%, donde Apache y Nginx son los dos servidores web de código abierto más usados.

Nginx se caracteriza por poseer una arquitectura orientada a eventos para el manejo de peticiones, presentar un sistema de módulos estático, funciona como proxy inverso y posee soporte para hosts virtuales basados en IP y/o hosts virtuales basados en nombre. Por otra parte Apache 2 se caracteriza por la introducción de Módulos de Multiprocesamiento (MPM, del inglés *Multiprocessing Modules*) en el manejo de peticiones, un sistema de módulos dinámico, funciona como servidor proxy caché y posee soporte para hosts virtuales basados en IP y/o hosts virtuales basados en nombre.

En Cuba también se evidencia el gran uso de estos dos servidores web de código abierto. En el Anexo 1 se muestra una tabla con los 40 principales sitios en Cuba de interés cultural y de entretenimiento, informativo, educativo e investigativo (ETECSA, 2018) y las tecnologías que estos emplean. Para analizar los sitios web se utilizó la herramienta multiplataforma Wappalizer, que permite identificar la tecnología utilizada en sitios web, incluidos servidores, sistemas de gestión de contenido, plataformas de comercio electrónico, herramientas de análisis, marcos de publicidad, etc. (RAKHMAWATI, et al., 2018). Del análisis realizado se determinó lo siguiente:

- 17 mostraron el sistema operativo, donde 88,2% se basan en código abierto (Ubuntu, Debian, RedHat, SUSE, CentOS) y 11,8% emplean Windows Server.
- 33 mostraron el servidor web, donde las estadísticas son Apache 2 (66,6%), Nginx (27,3%) y Microsoft-IIS (6,1%), evidenciándose que Apache 2 es notablemente el más usado.

- 28 mostraron el lenguaje de programación, PHP (89,3%), ASP.NET (7,1%) y Python (3,6%), destacándose PHP en la mayoría de los sitios.
- 25 sitios están desarrollados con el lenguaje de programación PHP, 21 de ellos muestran el servidor web, donde Nginx (28,6%) y Apache 2 (71,4%).

Se puede apreciar que se hace extensivo el uso de tecnologías de código abierto y precisamente en Cuba el proceso de informatización tiene como pilar el empleo de software libre. El 28 de febrero de 2017 quedó aprobada por el Consejo de Ministros la política integral para el perfeccionamiento de la informatización de la sociedad, donde se debe asegurar la sostenibilidad y soberanía tecnológica. Como precisó Miguel Díaz-Canel Bermúdez, al intervenir en la jornada previa al Noveno Período Ordinario de sesiones de la VIII legislatura de la Asamblea Nacional del Poder Popular, existe la necesidad de que las instituciones avancen en la implementación de la política de informatización e incorporen masiva y ordenadamente el uso de aplicaciones y sistemas informáticos desarrollados por las entidades cubanas (RODRÍGUEZ, 2017).

Para lograr la soberanía tecnológica y desempeñar el proceso de migración, el país creó una estructura compuesta por cuatro grupos de trabajo (Legal, Capacitación, Divulgación, Técnico) y el Grupo Ejecutivo, encargado de la dirección. Como parte del Grupo Técnico, la Universidad de las Ciencias Informáticas (UCI) se destaca por desarrollar la distribución cubana de GNU/Linux Nova y llevar a cabo el proceso de migración a tecnologías de software libre y plataformas de código abierto en diferentes instituciones cubanas (PÉREZ, 2015).

Nova utiliza el núcleo Linux e incluye determinados paquetes de aplicaciones informáticas para satisfacer las necesidades de la migración a plataformas de código abierto que experimenta Cuba como parte del proceso de informatización de la sociedad. Es el sistema operativo recomendado para ser utilizado como tecnología base de despliegue y desarrollo de aplicaciones informáticas en los Organismos de la Administración Central del Estado (OACE). En tal sentido, el sistema ha sido instalado en más de 100 000 computadoras ensambladas en la fábrica de GEDEME, que tienen como destino organismos e instituciones nacionales (UCI, 2018a). Una de las tres variantes es Nova Servidores, una distribución de GNU/Linux orientada a servidores usando estándares abiertos y adaptada a los entornos de las empresas cubanas (UCI, 2018b).

La UCI ha realizado varios procesos de migración en entidades como: Empresa Telemática del Ministerio de la Industria Pesquera (Telemar), Escuela Superior del Partido Comunista de Cuba “Ñico López”, Centro de Cibernética Aplicada a la Medicina (CECAM), la Empresa Constructora de Obras de Arquitectura No 24 (ECOA24) y la Empresa Constructora de Obras de Arquitectura e Industriales No.3 (ECOAIND3). En el año 2015 se realizó un diagnóstico en 10 ministerios con vísperas a una futura

migración a software libre y plataformas de código abierto, estos son: Ministerio de Educación Superior (MES), Ministerio de Educación (MINED), Ministerio de Industrias (MINDUS), Ministerio de Comunicaciones (MINCOM), Ministerio de la Construcción (MICONS), Banco Central de Cuba (BCC), Oficina Nacional de Estadísticas e Información (ONEI), Ministerio de Relaciones Exteriores (MINREX), Ministerio de Salud Pública (MINSAP) y Ministerio de Comercio Exterior (MINCEX).

La ejecución de los procesos de migración a código abierto inicialmente tenía como base la "Guía Cubana de Migración a Software Libre" confeccionada en la propia universidad en el año 2009. A partir del año 2015 se aplica la "Estrategia para la migración a aplicaciones de código abierto" elaborada por PÉREZ (2015), la cual corrige un conjunto de dificultades detectadas en la guía. La estrategia plantea tres etapas:

- Preparación: Donde se ejecutan todas las tareas de diagnóstico de los procesos, personas y tecnología de la entidad, se realizan tareas de análisis de factibilidad de la migración y se emite el plan de migración institucional.
- Ejecución: Comprende las actividades necesarias para la migración definitiva de usuarios y tecnologías de la institución. Incluye la migración de los servicios telemáticos y las computadoras de escritorio.
- Consolidación: Etapa que comprende tareas destinadas a garantizar el soporte técnico a los usuarios e infraestructura.

PÉREZ (2015) plantea que la migración de servicios telemáticos es el primer paso en el proceso de ejecución de la migración. Dentro de estos servicios se encuentran los ofrecidos por los servidores web. Se plantea que de cada servidor presente en la entidad debe obtenerse toda la información relacionada con el hardware, la información de los servicios brindados y la relación existente entre estos y los demás sistemas de la institución (PÉREZ, 2015).

Además como apoyo al proceso de migración se encuentra el libro "Buenas Prácticas para la Migración a Código Abierto", el cual establece que de manera análoga debe realizarse la selección de las aplicaciones informáticas que servirán de soporte para los servicios telemáticos de la institución. Plantea los elementos de gran importancia a tener en cuenta en la selección de las alternativas para la migración de los servicios, como son los conocimientos y experiencia de los administradores de red sobre el uso de las tecnologías libres. Por otra parte, la selección de la alternativa libre debe adecuarse a las necesidades de la institución y no a la alternativa al servicio privativo existente. El libro define que las alternativas libres a servidores web privativos para PYMES (Pequeñas y Medianas Empresas) y grandes empresas son Apache 2 y Nginx (PÉREZ, GARCÍA y GOÑI, 2015).

En estas tareas el protagonismo lo tiene el especialista en servicios telemáticos, que es responsable de planificar, diseñar y ejecutar la migración de los servicios telemáticos de la institución. Debe ser capaz de seleccionar la mejor variante para la migración de los servidores preferentemente de forma transparente a los usuarios así como realizar la migración de los datos existentes a la nueva infraestructura de servicios minimizando la pérdida de información. Debe garantizar además la seguridad informática de la solución de servicios propuesta (PÉREZ, 2015).

Para los servidores web se definen las dos alternativas pero no cómo se debe realizar el proceso de selección de estos. A partir de una entrevista realizada a los especialistas en servicios telemáticos pertenecientes a la UCI (ver Anexo 2), se identificó que para la selección no existe una guía que oriente qué servidor se ajusta más a la institución. No se tiene un registro del estudio de servidores web migrados en procesos anteriores. La situación existente ocasiona dificultades tales como:

- Se evidencia la pérdida de tiempo y se duplican esfuerzos estudiando la alternativa a seleccionar.
- No se reutiliza el conocimiento para la selección del servidor web.
- Muchas veces se selecciona el servidor web más conocido aunque sea el menos eficiente para la institución.

Dada la problemática existente, se define como **problema científico** de la investigación: ¿Cómo aumentar la eficiencia en la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto?

Con vista a la solución del problema planteado, el **objeto de estudio** está orientado a los servidores web Apache 2 y Nginx.

El **objetivo general** está centrado en desarrollar una solución informática para aumentar la eficiencia en la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto.

El **campo de acción** está enmarcado en el proceso de selección de los servidores web Apache 2 y Nginx. Los **objetivos específicos** plantean:

1) Construir el marco teórico de la investigación relacionado con los servidores web Apache 2 y Nginx.
2) Fundamentar la selección del servidor web a partir de la eficiencia de Apache 2 y Nginx.
3) Desarrollar la solución informática para la selección de los servidores web Apache 2 y Nginx.
4) Validar la solución informática para la selección de los servidores web Apache 2 y Nginx.

Se plantea como **hipótesis** que el desarrollo de una solución informática permitirá aumentar la eficiencia en la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto, identificándose como **variable independiente** la solución informática y como **variable dependiente** la eficiencia en la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto.

Se define **eficiencia** como la capacidad de alcanzar un objetivo determinado mediante una utilización racional de los recursos disponibles, sean materiales o tiempo, o sea, con el mínimo de recursos posibles y/o en el menor tiempo posible (PARTIDO COMUNISTA DE CUBA, 2017). La operacionalización de la variable dependiente se detalla en el Anexo 3.

Para el desarrollo de la investigación se utilizaron los **métodos científicos** y **técnicas** que a continuación se detallan.

- **Analítico-Sintético:** permitió el estudio de diferentes fuentes bibliográficas para extraer los elementos más importantes que se relacionan con los servidores web de código abierto Apache 2 y Nginx y a partir de los mismos determinar las particularidades de cada servidor.
- **Hipotético deductivo:** permitió verificar la hipótesis planteada en la investigación y establecer nuevas predicciones a partir de los conocimientos con los que ya se dispone.
- **Modelación:** permitió la creación de abstracciones con el objetivo de explicar la forma en la que cada servidor web procesa las peticiones que recibe. Se elaboraron representaciones explícitas de los tipos de arquitectura, la integración de Nginx como proxy inverso de Apache 2 y el funcionamiento del servidor web.
- **Estadístico:** permitió el procesamiento de la información recopilada acerca del comportamiento de Apache 2 y Nginx, facilitando las generalizaciones e interpretaciones de los datos. Se aplicó la estadística descriptiva utilizando una escala cuantitativa y la media como característica descriptiva, a los resultados de las pruebas de eficiencia aplicadas a los servidores web, con el objetivo de caracterizar los conjuntos de datos numéricos, poniendo de manifiesto de forma gráfica las propiedades de cada servidor.
- **Entrevista:** aplicada a los especialistas en servicios telemáticos del Centro de Software Libre para obtener información acerca de la experiencia en cuanto al proceso de selección de servidores web durante la migración a código abierto. Además se le aplicó a administradores de servidores web para conocer cómo se realiza actualmente el proceso de selección de estos en su institución.

- **Encuesta:** técnica de recopilación de información por medio de preguntas escritas organizadas en un formulario impreso, que se aplicó para determinar el nivel de satisfacción de los especialistas en servicios telemáticos con la solución propuesta como parte de la técnica de Iadov y en el marco del método Delphi para el criterio de expertos.

El **aporte social** de la presente investigación es que representa un resultado palpable de la implementación de la política de informatización de la sociedad cubana. Cumple con los principios aprobados en el 7mo Congreso del Partido Comunista de Cuba, asegurando la soberanía tecnológica e incorporando el empleo de una aplicación informática desarrollada por una entidad cubana.

Como **aporte práctico** una solución informática que permite la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto.

La investigación está estructurada en tres capítulos, los cuales se describen a continuación:

- **Capítulo 1. Servidores web Apache 2 y Nginx:** se realiza un estudio de los servidores web de código abierto Apache 2 y Nginx, determinando sus características y rasgos identificativos. Posteriormente se comparan ambos servidores destacando los aspectos que los diferencian. Se realiza un esquema del funcionamiento de los servidores web y se identifican los elementos que intervienen en el correcto desempeño de los mismos.
- **Capítulo 2. Solución para la selección de Apache 2 y Nginx:** con la realización de un estudio de caso se formula la teoría referente a la eficiencia de los servidores web Apache 2 y Nginx, a partir de la cual se definen 60 escenarios para la selección del servidor web más eficiente. Además se describe la concepción, arquitectura y funcionalidades de la solución informática que da cumplimiento al objetivo planteado en la presente investigación.
- **Capítulo 3. Evaluación de la solución:** a partir del desarrollo de la solución informática y los indicadores de eficiencia a medir, se realiza un estudio de caso en ECOAIND3 donde se evalúan los resultados midiendo los indicadores definidos. Además se aplica el criterio de expertos con el método Delphi y la técnica de Iadov para conocer el nivel de satisfacción de los especialistas en servicios telemáticos con la solución desarrollada. Finalmente se aplica la triangulación metodológica para verificar la convergencia de los métodos aplicados.

# CAPÍTULO 1. SERVIDORES WEB APACHE 2 Y NGINX

En el presente capítulo se realiza un estudio referente a los servidores web de código abierto Apache 2 y Nginx. Como resultado del mismo, se realiza una comparación entre ambos identificando las características que los diferencian y los elementos que intervienen en su correcto desempeño.

## 1.1 Servidores web

Los servidores generalmente son considerados como el hardware que lo alberga, pero el servidor en realidad es el software que realiza, controla o coordina un servicio o recurso (GILSTER, 2001). Por tanto, en la presente investigación la autora se refiere con el término servidor estrictamente al software. A aquellas computadoras en red que realizan una tarea especial para atender las necesidades de recursos de las estaciones de trabajo (clientes) en una red (GILSTER, 2001), la autora de la investigación las considera como PC servidoras.

Un servidor web es una aplicación que responde a solicitudes provenientes de navegadores web, proporcionando recursos solicitados a través del protocolo HTTP o de manera segura a través del protocolo HTTPS (MEJÍA, GONZÁLES y ESPAÑA, 2018). Cuando una solicitud viene del cliente al servidor, toma esa solicitud de los usuarios, la procesa y envía la respuesta correspondiente (NANDAL, et al., 2018). Un servidor web brinda las respuestas HTTP, generalmente en forma de páginas web que contienen contenido estático (texto, imágenes, etc.) y dinámico (scripts) (KUNDA, CHIHANA y SINYINDA, 2017).

### 1.1.1 Elementos de los que depende el correcto desempeño

Cada acción en el sistema cuesta ciclos de la unidad central de procesamiento (CPU, por sus siglas en inglés *Central Processor Unit*), la mayoría requiere memoria, el intercambio de información ocupa ancho de banda, los datos ocupan espacio de almacenamiento y la comunicación entre dispositivos consume el rendimiento de entrada/salida (E/S). Los patrones de uso de recursos cambian con la carga. El monitoreo completo debe cubrir tres grupos principales de métricas: disponibilidad de recursos, rendimiento del software y, cuando corresponda, comportamiento del usuario (LIGUS, 2013).

En los servidores web si bien las características y los asuntos relacionados con la comunidad son importantes en general, el aspecto que puede marcar la diferencia es el rendimiento (NEDELCU, 2015). Una acción computacional típica en un entorno de servicio web es una solicitud. Cada solicitud requiere recursos, como mínimo consume memoria y ciclos de CPU, pero con frecuencia también lee y escribe algunos datos en otros dispositivos, como unidades de disco, lo que introduce un mayor costo de E/S. El agotamiento de cualquier recurso requerido para atender una solicitud lleva a la creación del llamado cuello de botella de rendimiento. Por lo general, no existe una respuesta única a las causas de los altos

tiempos de respuesta, pero la mayor parte del tiempo implica una escasez de recursos subyacentes, ya sea ancho de banda de red, CPU, E/S o la Memoria de Acceso Aleatorio (RAM, del inglés *Random Access Memory*). Al mismo tiempo, las solicitudes de larga duración aún ocupan recursos que podrían utilizarse mejor si las solicitudes tienen éxito (LIGUS, 2013).

El éxito de un servidor web depende en gran medida de su rendimiento, el cual es un elemento importante en el posicionamiento eficaz de los sitios web. La posibilidad de que se carguen rápido las páginas, exista estabilidad en el contenido mostrado y se soporte gran concurrencia de usuarios, influye en que la experiencia del visitante se torne más agradable y productiva (HERNÁNDEZ, 2014). Es importante tener en cuenta que las solicitudes con tiempos de respuesta extremadamente largos tienen el mismo efecto que la pérdida de disponibilidad: los usuarios impacientes simplemente se rendirán. Los administradores tienden naturalmente a favorecer al servidor que proporcionará una comodidad óptima para el usuario final, que se caracteriza por tiempos de carga de página mínimos y velocidades máximas de descarga (NEDELCU, 2015).

La prestación exitosa de los servicios de un servidor web depende de su rendimiento, el cual es la proporción entre el producto o el resultado obtenido y los medios utilizados (REAL ACADEMIA ESPAÑOLA, 2018). Según KABIR (2003), la fórmula matemática más sencilla posible para el rendimiento de un servidor web es *Rendimiento = (hardware, software, red, contenido)* que significa que el rendimiento está en función del hardware, del software, de la red y del contenido (KABIR, 2003). El rendimiento del servidor web es fundamental para la comunicación de información efectiva y eficiente, se refiere a la capacidad para responder a las solicitudes de los usuarios (KUNDA, CHIHANA y SINYINDA, 2017). Para los autores KUNDA, CHIHANA y SINYINDA (2017) las medidas de rendimiento incluyen utilización de recursos como memoria y CPU, el disco local, el tiempo de respuesta, entre otros, todos utilizando archivos estáticos y archivos dinámicos. A continuación se describen estos elementos.

**Unidad Central de Procesamiento:** procesador es una abreviatura de microprocesador y de unidad central de procesamiento, es un sistema de circuitos electrónicos que utiliza la lógica digital para realizar instrucciones del software (GILSTER, 2001). Continuamente sigue el ciclo Extraer (recupera una instrucción de la memoria principal), Decodificar (determina cuál es la instrucción) y Ejecutar (llevar a cabo la instrucción) (LEWIS, 2009):

**Memoria de Acceso Aleatorio:** la RAM se utiliza en la computadora para su memoria primaria o principal, es donde se almacenan todos los programas activos y datos de manera que están disponibles rápidamente y se tenga acceso a ellos fácilmente por parte de la CPU y otros componentes de la computadora. Tener más RAM en la computadora no mejora la velocidad del procesador, pero sí mejora

la cantidad de datos a los que el procesador puede tener acceso sin necesidad de ir a la unidad de disco duro que es más lento (GILSTER, 2001).

Un elemento importante es el área de intercambio (swap), si una tarea comienza utilizando mucha memoria, el host se verá obligado a liberar memoria al poner otras tareas en la memoria de intercambio. Acceder a la memoria de intercambio, que después de todo es solo una pieza de disco que se usa como si fuera memoria, es muy lenta en comparación con el acceso a la RAM. El espacio de intercambio y la RAM conforman la memoria asignable total disponible para el sistema (MATOTEK, TURNBULL y LIEVERDINK, 2017).

**Disco duro:** un servidor web gasta cierta cantidad de tiempo accediendo al disco duro, lo que puede ser una causa de pérdida de rendimiento (KABIR, 2003). El espacio en disco y la velocidad del disco son unos de los recursos finitos que tiene un sistema informático. Un sistema no se ralentizará cuando un disco se llena, pero los servicios pueden fallar y causarle problemas a los usuarios, por lo que se debe vigilar el uso y tener más espacio de almacenamiento disponible cuando sea necesario (MATOTEK, TURNBULL y LIEVERDINK, 2017).

**Ancho de banda:** es el número de datos en bits, normalmente kilobits o megabits, que un cable puede transmitir en un segundo. Por ejemplo, el cable UTP se clasifica nominalmente a 10 Mbps, o diez millones de bits por segundo (GILSTER, 2001).

**Tipo de contenido:** con la web 1.0 solo existían páginas estáticas, sin embargo con la introducción de la web 2.0 se introducen las páginas dinámicas con el empleo de CGI (del inglés *Common Gateway Interface*), PHP (del inglés *Hypertext Preprocessor*), Perl, Python[1], entre otros. Por tanto existen dos tipos de contenido: estático y dinámico (NEDELCU, 2015):

- Archivos estáticos: todo el contenido que no requiera procesamiento adicional, como imágenes, archivos CSS (del inglés *Cascading Style Sheets*), archivos HTML (del inglés *HyperText Markup Language*) estáticos (NEDELCU, 2015), archivos JavaScript, texto, PDF, etc. (SONI, 2016).
- Archivos dinámicos: requieren procesamiento antes de ser enviados al cliente, como los scripts PHP, Python, Perl y Ruby (NEDELCU, 2015). Los sitios dinámicos tienen scripts y lenguajes de programación que funcionan en la parte posterior y emiten páginas que el navegador puede comprender y representar directamente (SONI, 2016).

**Tiempo de respuesta:** es el tiempo que tarda la aplicación en responder a la solicitud del usuario. Se mide en segundos o milisegundos según sea apropiado para la aplicación (MATAM y JAIN, 2017).

[1] Python es un lenguaje de programación dinámico y de propósito general que se utiliza en varios campos, en la programación web del lado del cliente y del servidor (EMBARAK, 2018).

Los autores KUNDA, CHIHANA y SINYINDA (2017) realizaron una comparación de rendimiento de los servidores web con diferentes arquitecturas, donde obtuvieron como resultado que los servidores web con arquitectura asíncrona[2] pueden lograr una capacidad de respuesta mucho mejor que la versión basada en hilos debido a su robustez para manejar una alta carga de trabajo de concurrencia. A continuación se describen los tipos de arquitectura de un servidor web.

### 1.1.2 Arquitectura para el manejo de peticiones

La arquitectura de software de un servidor web proporciona un servicio que establece cómo un usuario accede a través del protocolo HTTP a los archivos de datos almacenados en el servidor web. Existen tres arquitecturas: basadas en procesos, basadas en hilos[3] y basadas en eventos. En las dos primeras la E/S se bloquea durante el manejo de la solicitud, por tanto el consumo de recursos se incrementa linealmente de acuerdo con el número creciente de hilos o procesos. La tercera es mucho más flexible que las anteriores, con E/S sin bloqueo y manejo asíncrono de solicitudes (NAM, 2017).

#### Servidores basados en procesos

Es el predecesor de los demás diseños. Cuando se crea un nuevo proceso, el sistema duplica la imagen del ejecutable en un nuevo espacio de direccionamiento y, por lo tanto, a partir de ese punto los procesos son independientes (CHARTE, 2003). Un proceso es una instancia específica de un programa de computadora en ejecución. Los servidores basados en procesos son excelentes porque son estables y el bloqueo de un proceso no afecta a otros procesos. Sin embargo, no pueden manejar tantos clientes porque la creación y destrucción de todos los procesos crea una gran cantidad de sobrecarga del procesador y requiere una gran cantidad de memoria (HELMKE, 2015). El uso de la memoria, en este caso, es linealmente proporcional al número de procesos secundarios. Además, el sistema operativo necesita gastar más tiempo de CPU para cambiar entre los procesos secundarios (cambio de contexto) (NAM, 2017).

#### Servidores basados en hilos

Al crear un nuevo hilo de ejecución, el sistema no crea un nuevo proceso ni duplica la imagen del ejecutable, sino que, sobre el mismo espacio de direcciones y compartiendo la misma área de memoria, pone en marcha un nuevo puntero de instrucción y comienza a ejecutar sentencias (CHARTE, 2003). Un proceso puede controlar muchos hilos, esto es bueno para la administración de recursos. Mediante el uso de hilos en lugar de procesos, se puede atender un mayor número de solicitudes de clientes

[2] Una comunicación es asíncrona cuando el envío y la recepción de información entre un remitente y un receptor no ocurren necesariamente en el mismo instante (CACCIAGRANO y CORRADINI, 2001).

[3] Un hilo, hebra o thread es un recurso del sistema capaz de ejecutar un código alojado en el espacio de direccionamiento de un cierto proceso, pudiendo existir más de un hilo asociado a un mismo proceso (CHARTE, 2003).

utilizando menos recursos del sistema que un servidor basado en procesos. En este tipo de servidores, las solicitudes pueden compartir memoria, haciéndolas más eficientes y por lo tanto capaces de atender más solicitudes de forma más rápida, sin embargo, un bloqueo en un hilo podría hacer que todo el servidor caiga (HELMKE, 2015).

**Servidores basados en eventos**

En la arquitectura controlada por eventos, un bucle de eventos de un solo proceso de un solo hilo ejecuta múltiples conexiones mediante mecanismos de E/S no bloqueantes. Puede descargar el cuello de botella en el rendimiento en caso de una alta carga de conexión gracias a su comunicación asíncrona. La arquitectura basada en eventos puede atender más solicitudes concurrentes que la basada en procesos o en hilos (NAM, 2017).

Esta arquitectura basa su funcionamiento en la utilización de lecturas y escrituras asíncronas sobre sockets. En lugar de iniciar un nuevo proceso o un nuevo hilo para cada solicitud, en una arquitectura impulsada por eventos, el flujo del programa está controlado por eventos, alguna forma de mensaje enviado desde otro proceso o hilo. Aquí, tiene un proceso que se ejecuta como un “detector” o “detector de eventos” que espera que una solicitud ingrese al servidor. Cuando llega la solicitud, en lugar de comenzar un nuevo proceso, se envía un mensaje a una parte diferente del servidor llamada “controlador de eventos”, que luego realiza una tarea. El resultado simplificado de servir páginas web de esta manera es que se necesita menos tiempo de procesador y menos memoria (HELMKE, 2015).

Actualmente la mayoría de los diseños de servidores web de alto rendimiento implementan arquitecturas híbridas. Apache proporciona una arquitectura flexible basada en procesos y basada en hilos, mientras que Nginx presenta una potente arquitectura multiproceso e impulsada por eventos (NAM, 2017). A continuación se describen ambos servidores.

## 1.2 Apache 2

Apache, lanzado en el año 1994, es un servidor web de código abierto cuya implementación se realiza de forma colaborativa. El proyecto está dirigido y controlado por un grupo de voluntarios de todo el mundo conocido como el Grupo Apache. Apache 2 es una profunda revisión del servidor Apache, centrándose en la escalabilidad, seguridad y rendimiento. Las principales revisiones de código se han llevado a cabo para crear una arquitectura realmente escalable (KABIR, 2003).

### 1.2.1 Arquitectura para el manejo de peticiones

En Apache 2 cada solicitud es atendida por un hilo separado o proceso y utiliza sockets sincrónicos[4] (NEDELCU, 2015). Con la creación de la versión 2.0 se introducen los MPM, como una solución propuesta por el Grupo Apache debido a que la anterior arquitectura no trabajaba bien bajo plataformas que no estaban centradas en procesos como en el caso de Windows (KABIR, 2003). Los MPM son responsables de conectar con los puertos de red de la PC servidora, aceptar las peticiones y generar los procesos hijos que se encargan de servirlas. Existen diferentes MPM, encontrándose entre ellos *prefork, worker, event, threaded* y *perchild,* cada uno basado en un funcionamiento diferente. Los sitios web que necesitan más que nada escalabilidad pueden usar un MPM hebrado como *worker*, mientras que los sitios web que requieran por encima de otras cosas estabilidad o compatibilidad con software antiguo pueden usar *prefork*. Además, se pueden configurar funcionalidades especiales como servir diferentes hosts con diferentes identificadores de usuario con el uso de *perchild* (APACHE SOFTWARE FOUNDATION, 2017). A continuación se describen los tres MPM que trae Apache 2 por defecto.

**MPM *prefork***

Este MPM crea un grupo de procesos hijos para servir peticiones. Cada proceso hijo tiene un solo hilo, por lo que si Apache inicia 20 procesos hijos, puede servir 20 peticiones simultáneamente, como se aprecia en la Figura 1. En caso que ocurra un error grave y un proceso hijo muera, solo se pierde una petición: la que está atendiendo el hijo que acaba de cesar la actividad. El máximo y el mínimo de procesos hijos son parámetros configurables, permitiendo adaptar Apache a las capacidades de la PC servidora y también fijarle un máximo de carga. Dentro de estos parámetros máximo y mínimo el servidor genera más o menos procesos en función de la carga de trabajo actual.

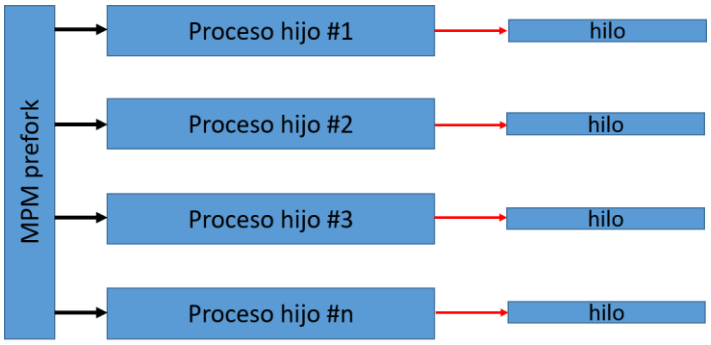


*Figura 1: Funcionamiento del MPM prefork (Fuente: elaboración propia).*

En el MPM *prefork* el rendimiento se degrada rápidamente después que las solicitudes superen el número de procesos, por lo que esta no es una buena opción en muchos escenarios. Cada proceso

[4] Una comunicación es sincrónica cuando el envío y la recepción de información entre un remitente y un receptor son eventos simultáneos (CACCIAGRANO y CORRADINI, 2001).

tiene un impacto significativo en el consumo de RAM, por lo que este MPM es difícil de escalar de manera efectiva. Puede ser una buena opción si se usa junto con otros componentes que no están construidos con hilos como PHP (el cual no es seguro para hilos), por lo que este MPM se recomienda como la única forma segura de trabajar con *mod_php*, el módulo de Apache para procesar estos archivos (CAMPOVERDE, HERNÁNDEZ y MAZÓN, 2015).

**MPM *worker* y MPM *event***

El MPM *event* es una mejora de *worker* pero se basa en el mismo funcionamiento multiproceso-multihilo[5], como se aprecia en la Figura 2. Usan hilos para atender peticiones, el servidor puede servir un mayor número de peticiones con menos recursos del sistema que un servidor basado únicamente en procesos. Si Apache inicia 20 procesos hijos y cada uno 30 hilos, se pueden servir 600 peticiones simultáneamente. El MPM *event* es similar a *worker* en la mayoría de las situaciones, pero está optimizado para manejar conexiones keep-alive (CAMPOVERDE, HERNÁNDEZ y MAZÓN, 2015).

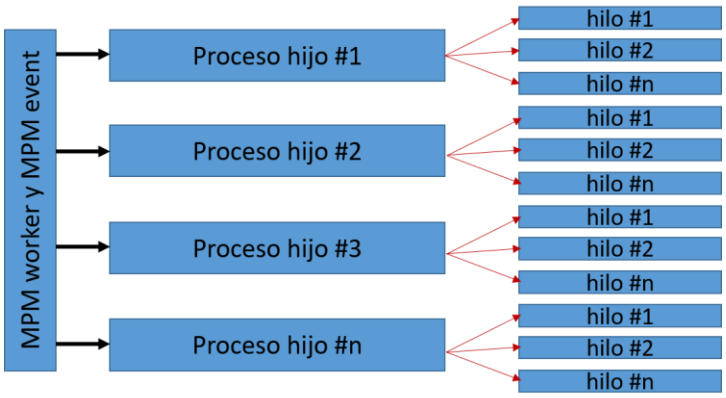


*Figura 2: Funcionamiento de los MPM worker y event (Fuente: elaboración propia).*

Tanto en *worker* como *event*, hay dos opciones de rendimiento principales que se pueden ajustar. El primero es *ThreadsPerChild*, y el segundo es *MaxRequestWorkers*. *ThreadsPerChild* describe cuántos hilos se crean desde cada proceso hijo, mientras que *MaxRequestWorkers* determina los hilos máximos lanzados en total. Una sola conexión web de un usuario puede lanzar más de un hilo. Cada conexión de usuario aumenta la cantidad de recursos del sistema, como la RAM y el tiempo de CPU (MATOTEK, TURNBULL y LIEVERDINK, 2017).

### 1.2.2 Soporte para protocolos basados en CGI

Apache posee soporte para CGI y FastCGI (del inglés *Fast Common Gateway Interface*), a través de módulos que se pueden cargar en el servidor (NEDELCU, 2015).

[5] Aplicaciones multihilo, multihebra o multithread son aquellas que, desde el hilo de ejecución principal que se pone en marcha al iniciar el proceso, crean hilos adicionales según los van necesitando (CHARTE, 2003).

**Protocolo CGI**

Cuando un cliente intenta visitar una página dinámica, el servidor web recibe la solicitud y la reenvía a una aplicación de terceros. La aplicación procesa el script de forma independiente y devuelve la respuesta producida al servidor web, que luego reenvía la respuesta al cliente. Para que el servidor web se comunique con esa aplicación, el protocolo CGI se inventó a principios de la década de 1990, permite que un servidor HTTP y un script CGI compartan la responsabilidad de responder a las solicitudes de los clientes. El servidor es responsable de gestionar los problemas de conexión, transferencia de datos, transporte y red relacionados con la solicitud del cliente, mientras que el script CGI maneja los problemas de la aplicación, como el acceso a datos y el procesamiento de documentos. CGI es el protocolo que describe la forma en que se intercambia la información entre el servidor web y la aplicación de puerta de enlace (PHP, Python, etc.). Si bien esta tecnología parece lo suficientemente simple y eficiente al principio, viene con algunos inconvenientes principales (NEDELCU, 2015):

- Se genera un proceso único para cada solicitud. La memoria y otra información de contexto se pierden de una solicitud a otra.
- Iniciar un proceso puede consumir muchos recursos para el sistema. Cantidades masivas de solicitudes simultáneas (cada vez que se genera un proceso) podrían saturar rápidamente un servidor.

**Protocolo FastCGI**

El protocolo CGI es relativamente ineficiente para los servidores que están sujetos a grandes cargas. A mediados de los años 90, la voluntad de encontrar soluciones llevó a Open Market a desarrollar una evolución de CGI: FastCGI. Este es un protocolo de comunicación que se ejecuta a través de sockets, lo que implica que hay un cliente y un servidor. Aunque el objetivo sigue siendo el mismo, FastCGI ofrece mejoras significativas sobre CGI. En lugar de engendrar un nuevo proceso para cada solicitud, emplea procesos persistentes que vienen con la capacidad de manejar solicitudes múltiples. El servidor web y la aplicación de puerta de enlace se comunican con el uso de sockets, en consecuencia, los procesos del servidor web y del back-end pueden ubicarse en dos computadoras diferentes en una red. El servidor web reenvía la solicitud del cliente a la puerta de enlace y recibe la respuesta dentro de una única conexión. También pueden seguir solicitudes adicionales sin necesidad de crear conexiones adicionales. En la mayoría de los servidores web, incluidos Nginx y Apache, la implementación de FastCGI no admite (o al menos no totalmente) la multiplexación. Como FastCGI es un protocolo basado en socket, puede implementarse en cualquier plataforma con cualquier lenguaje de programación (NEDELCU, 2015).

### 1.2.3 Principales características

Apache 2 cuenta con características importantes como:

**Multiplataforma**: es utilizado en multitud de sistemas operativos, lo que lo hace prácticamente universal. Actualmente funciona en Windows, GNU/Linux, Unix, BSD, Mac OS X, Solaris, Novell NetWare, OS/2, TPF, OpenVMS, eCS, AIX, z/OS, HP-UX, entre otros (NEDELCU, 2015).

**Sistema de módulos dinámico**: Apache utiliza el concepto de objetos compartidos dinámicos (DSO, del inglés *Dynamic Shared Object*) en el que puede cargar o descargar módulos en tiempo de ejecución después de compilar Apache. Usando DSO, los módulos no están incluidos en el proceso principal de Apache y por lo tanto, le permite cargar o descargar módulos dinámicamente (SONI, 2016). Apache puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona. Existen infinidad de módulos disponibles, todos ellos denominados mod_XXX, donde el nombre XXX designa el tipo de funcionalidad que está pensado que desempeñen. Esta orientación permite al servidor ser personalizado de forma muy precisa, instalando o desinstalando servicios a demanda de los usuarios del mismo y que de esta forma solo estén en marcha aquellas funcionalidades realmente necesarias (PALMA, 2013).

**Soporte de hosts virtuales**: Apache es uno de los primeros servidores web en soportar hosts virtuales basados en IP y/o hosts virtuales basados en nombre. El servidor Apache puede ejecutarse como un único servidor web para un único sitio o puede servir a cientos de sitios como hosts virtuales. De esta manera, muchos sitios web comparten los recursos subyacentes, como la CPU, los recursos de disco y las direcciones de Protocolo de Internet (IP) de un único servidor web. El alojamiento virtual basado en IP hace que Apache sirva una página web desde un directorio específico, según la dirección IP en la que se recibió la solicitud. Para cada host virtual basado en IP, el host Linux necesita tener una dirección IP asignada a una interfaz de red. Esto se hace agregando direcciones adicionales a una interfaz. El alojamiento virtual basado en nombre hace que Apache sirva una página web para un directorio específico basado en el nombre del sitio al que se conectó un usuario remoto. Cualquier número de hosts virtuales basados en nombre puede compartir una sola dirección IP. El nombre del sitio está determinado por un encabezado especial que se envía en la solicitud al servidor web (MATOTEK, TURNBULL y LIEVERDINK, 2017).

**Servidor proxy integrado:** un servidor proxy se encuentra entre las PC de los clientes e Internet, realmente es solo una capa adicional entre el cliente y el servidor (HELMKE, 2015). En términos generales, un servidor proxy actúa como un servidor para los host del cliente y como un cliente para los servidores remotos. Apache se puede convertir en un servidor proxy caché, el cual es un servidor intermedio que se sitúa entre el cliente y el servidor que tiene los contenidos. En este proceso se emplea

el módulo *mod_proxy*. Cuando un host solicita un cierto recurso remoto utilizando una URL, el servidor proxy recibe esta solicitud y utiliza el recurso para satisfacer la solicitud del cliente. El servicio proxy es perfecto para escenarios en los que están accediendo a la red varios usuarios. Este proceso permite al servidor proxy almacenar el contenido solicitado en un caché. Cualquier solicitud nueva que requiera información que se encuentra en el caché no necesita ser servida trayendo la información desde el servidor remoto. En lugar de eso, la nueva solicitud se sirve desde los datos del caché. Este tipo de servidores puede reducir retrasos en los tiempos de respuesta, conservar el ancho de banda y además ayuda a reducir el gasto general de sus comunicaciones. En los servidores proxy caché los usuarios pasan sus solicitudes al servidor proxy y este obtiene la respuesta del host objetivo de la solicitud. Los proxy cache están normalmente definidos explícitamente en los programas usuarios, como en el navegador web (KABIR, 2003).

**Requerimientos de hardware:** si el servidor sirve páginas estáticas mayoritariamente, la CPU no va a constituir un factor significativo en el rendimiento. Por el contrario, si genera una gran cantidad de contenido dinámico utilizando SSI (del inglés *Server Side Includes*), scripts CGI, y similares, el servidor Apache va a necesitar un procesador rápido. Sin embargo, una CPU rápida no significa una ventaja cuando se va a servir un alto volumen de contenido dinámico. El candidato más probable para los cuellos de botella, en ese tipo de escenarios es la RAM (KABIR, 2003).

### 1.2.4 Instalación y configuración

Apache se instala en el directorio */etc/apache2* con la ejecución del comando *apt-get install apache2*. Este servidor web no posee una interfaz de usuario gráfica para su administración, se trata de un archivo de configuración llamado *apache2.conf* o *httpd.conf*, según el sistema operativo, en este caso es apache2.conf debido a que se trabajará con el sistema operativo Nova Servidores. Se pueden mover partes de la configuración de Apache a diversos ficheros. Un ejemplo es poder aislar la configuración de cada host virtual en un archivo independiente para cada una, o bien distribuir la configuración de diferentes localizaciones físicas a ficheros independientes más pequeños y fáciles de manejar que se encuentren directamente en dichas localizaciones. No es necesario concentrar toda la configuración en un solo archivo de gran tamaño. De esta forma se evita que el fichero de configuración principal *apache2.conf* alcance un tamaño demasiado grande y por tanto sea más complejo de manejar (KABIR, 2003). El directorio */etc/apache2* incluye (PALMA, 2013):

- conf-available: directorio que contiene los ficheros de configuración disponibles.
- conf-enabled: directorio que contiene los ficheros de configuración activos.
- *mods-available*: directorio con los módulos disponibles, incluyendo los MPM. Cada módulo consta de un fichero para cargar (.load) y otro, opcional, para la configuración (.conf). Los

ficheros binarios de los módulos (.so) están en */usr/lib/apache2/modules*.

- *mods-enabled*: directorio con los módulos activos.
- *sites-available*: directorio con los ficheros de configuración de los hosts virtuales disponibles. Se recomienda crear cada host virtual en un fichero independiente.
- *sites-enabled*: directorio con los ficheros de configuración de los hosts virtuales activos. Se usa la misma filosofía que para activar módulos. Si se quiere activar un host virtual se crea un enlace directo en esta carpeta que apunte a *sites-available*.
- *apache2.conf*: fichero principal de configuración de Apache 2.
- *envvars*: fichero que incluye las variables de entorno que se deseen cargar.
- *magic*: datos para el módulo *mod_mime_magic*.
- *ports.conf*: fichero para configurar el puerto de comunicación HTTP.

La configuración se realiza mediante directivas y secciones. Las directivas son variables almacenadas en el archivo de texto de configuración para alterar y controlar el funcionamiento de Apache en tiempo de ejecución según sus valores y después de haber reiniciado el proceso servidor. Tienen la siguiente estructura: *NombreDeDirectiva ValorDeDirectiva*. Se aplican a todas las páginas del servidor pero si se quiere que ciertas de ellas se apliquen a páginas, directorios o ficheros concretos, se tendrán que usar secciones de configuración. En el archivo principal se pueden incluir diferentes ficheros de configuración mediante la directiva *Include*. Cada vez que se realice una modificación es necesario reiniciar o recargar Apache 2 para que los cambios tengan efecto (PALMA, 2013).

Algunos parámetros son generales para el servidor, mientras que otros se pueden configurar de manera independiente para cada conjunto de directorios o ficheros o para un host virtual concreto. La configuración de Apache está dividida en tres secciones fundamentales, aunque las directivas de cada una pueden aparecer mezcladas y desordenadas, estas son (PALMA, 2013):

- *Sección 1*: entorno global. Describe el funcionamiento general del servidor. Incluye parámetros como los puertos, MPM y manejo de conexiones, en algunos casos el fichero principal contiene la configuración y en otros la ruta a otro fichero.
- *Sección 2*: servidor principal. Se describe la configuración del servidor principal, que es la base sobre la que se construyen los hosts virtuales. La configuración está contenida en el archivo principal.
- *Sección 3*: hosts virtuales. Se puede alojar más de un host virtual en el mismo servidor, cada uno en un fichero de configuración independiente. En el archivo principal aparece la ruta al

directorio que contiene todos estos ficheros.

### 1.3 Nginx

Nginx es una creación de Igor Sysoev. Su desarrollo comenzó alrededor de 2002, y se lanzó públicamente en 2004. El equipo de Nginx lo describe como "*un servidor HTTP libre, código abierto, de alto rendimiento y proxy inverso, así como un servidor proxy IMAP/POP3*". Nginx es conocido por su alto rendimiento, estabilidad, amplio conjunto de funciones, configuración simple y bajo consumo de recursos. Nginx ha resuelto el problema de rendimiento y esa es una de las principales razones de todos los elogios y premios que conlleva. Es extremadamente rápido y brilla incluso bajo mucha carga. Nginx en realidad nunca comenzó como un servidor para lenguajes dinámicos como PHP (SONI, 2016).

Nginx evolucionó desde un simple experimento con la idea de resolver el problema C10k (SONI, 2016). El diseño original de Nginx se creó para permitir un mayor número de solicitudes simultáneas de sitios web, el problema de servir al menos a 10.000 clientes simultáneos desde un servidor web. Los sitios web más grandes suelen tener decenas de miles de clientes conectados simultáneamente, cada uno haciendo solicitudes HTTP a las que se debe responder. Los diseñadores de Nginx escucharon este problema, descrito como C10K, y decidieron que podían escribir un servidor web que fuera capaz de servir al menos a 10 000 clientes simultáneamente (HELMKE, 2015).

#### 1.3.1 Arquitectura para el manejo de peticiones

Nginx utiliza una arquitectura basada en eventos, una única instancia de Nginx consta de un proceso maestro y procesos de trabajo, como se aprecia en la Figura 3. Cada proceso de trabajo maneja conexiones múltiples, esto se logra ejecutando un bucle de eventos que extrae eventos que ocurrieron en sockets abiertos desde el sistema operativo. El sistema operativo es el encargado de distribuir las conexiones entrantes entre los procesos de trabajo de forma rotatoria (KHOLODKOV, 2015). Desde el archivo de configuración se puede definir la cantidad de procesos, las conexiones máximas por proceso y otros parámetros (PALMA, 2013).

Cada proceso de trabajo en Nginx tiene un solo hilo y se ejecuta de forma independiente. Su trabajo principal es obtener nuevas conexiones y procesarlas lo más rápido posible. Cuando se inician los procesos de trabajo, se inicializan con la configuración y el proceso maestro les dice que escuchen los sockets configurados. Una vez activos, leen y escriben contenido en el disco y se comunican con los servidores ascendentes. Como están todos bifurcados del proceso maestro, pueden usar la memoria compartida para datos en caché, datos de persistencia de sesión y otros recursos compartidos. En Windows, un hilo es comparativamente mucho más liviano que un proceso, a diferencia de Linux, donde sincronizar la memoria compartida es costoso, por eso en Nginx decidieron no crear múltiples hilos por proceso. El proceso maestro lee y evalúa los archivos de configuración, y también mantiene los

procesos de trabajo. Es el proceso de trabajo el que hace el procesamiento real de las solicitudes (SONI, 2016).

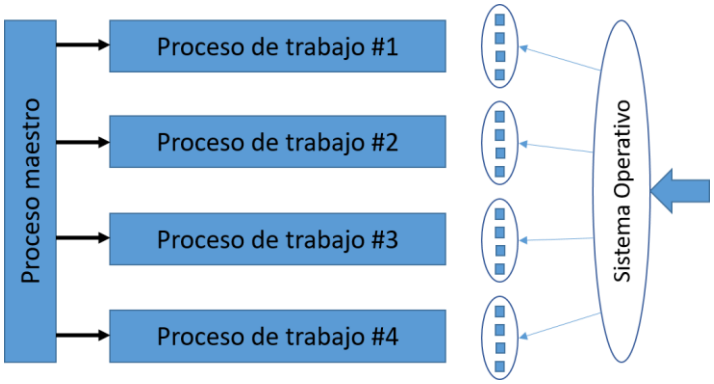


*Figura 3: Funcionamiento de la arquitectura de Nginx (Fuente: KHOLODKOV, 2015).*

Nginx es más eficiente que sus competidores, primero y ante todo, es más rápido al hacer uso de sockets asincrónicos. Nginx no genera procesos tantas veces como recibe solicitudes. Un proceso por núcleo es suficiente para manejar miles de conexiones, lo que genera una carga de CPU y un consumo de memoria mucho más ligeros. La razón por la cual Nginx está muy por delante de Apache en términos de rendimiento es porque es precisamente la razón por la que fue escrito (NEDELCU, 2015).

### 1.3.2 Soporte para protocolos basados en CGI

Nginx admite FastCGI y además ofrece otras dos implementaciones de módulos derivados de CGI: uWSGI y SCGI, a través de módulos que se incluyen por defecto en tiempo de compilación.

#### Protocolo uWSGI

El módulo uWSGI permite a Nginx comunicarse con las aplicaciones a través del protocolo uwsgi, que se deriva de (WSGI, del inglés *Web Server Gateway Interface*). El servidor más comúnmente usado (si no es el único) que implementa el protocolo uwsgi es el servidor uWSGI con un nombre no original. Este módulo es útil para Python, ya que el proyecto uWSGI fue diseñado principalmente para aplicaciones Python (NEDELCU, 2015).

#### Protocolo SCGI

SCGI (del inglés *Simple Common Gateway Interface*) es una variante del protocolo CGI, muy parecido al FastCGI. Es más joven que FastCGI ya que su especificación se publicó por primera vez en 2006, SCGI fue diseñado para ser más fácil de implementar y como su nombre lo sugiere: simple. No está relacionado con un lenguaje de programación particular. Las interfaces y módulos SCGI se pueden encontrar en una variedad de proyectos de software como Apache, IIS, Java, Cherokee y mucho más (NEDELCU, 2015).

### 1.3.3 Principales características

Nginx cuenta con características importantes como:

**Multiplataforma:** inicialmente creado para funcionar en sistemas operativos Unix, más tarde apareció una versión compatible con Windows, y quedó listo para trabajar en Linux, Unix, Windows, MAC OS, Solaris y BSD (NEDELCU, 2015).

**Sistema de módulos estático:** los módulos no pueden ser cargados de forma dinámica, deben ser incluidos en la compilación. Además, no se pueden desactivar en tiempo de ejecución debido a que están compilados e integrados en el binario principal (NEDELCU, 2015).

**Soporte de hosts virtuales:** posee soporte para hosts virtuales basados en IP y/o hosts virtuales basados en nombre.

**Proxy inverso:** el trabajo de un servidor proxy es enviar el pedido por adelantado y pasarlo al servidor proxy, que también se conoce como un servidor en sentido ascendente. El servidor ascendente procesa la solicitud y envía la respuesta nuevamente al servidor Nginx, que además retransmite la respuesta a los clientes que hicieron la solicitud. Nginx puede ser utilizado como proxy inverso con opciones de caché. Nginx puede hacer fácilmente el trabajo de front-end y enrutar el tráfico según sus requisitos al back-end utilizando sus capacidades de proxy inverso (SONI, 2016). Cuando se utiliza un servidor proxy inverso, los usuarios no son conscientes de su existencia porque piensan que están accediendo al recurso directamente. Todas las solicitudes que realiza el usuario se envían al proxy inverso, que sirve la respuesta desde su caché o recogiendo información de otro host (KABIR, 2003).

**Requerimientos de hardware:** en Nginx se genera una carga de CPU y un consumo de memoria mucho más ligeros (NEDELCU, 2015).

Otras características de Nginx son:

- Tiene soporte para la Capa de Sockets Seguros (SSL, del inglés *Secure Sockets Layer*) y la Seguridad de la Capa de Transporte (TLS, del inglés *Transport Layer Security*), streaming de video, autenticación.
- El soporte nativo para caché de datos y la posibilidad de configurar un balanceo de cargas entre los diferentes servidores que se tienen contratados.
- Incluye servicios de correo electrónico con acceso al Protocolo de Acceso a Mensajes de Internet (IMAP, del inglés *Internet Message Access Protocol*) y al Protocolo de Oficina de Correo (POP, del inglés *Post Office Protocol*).
- Manejo de archivos estáticos, archivos de índices y autoindexado.

- Posee el módulo de reescritura de URL.
- Control de acceso basado en la dirección IP del cliente y la autenticación básica de acceso HTTP.
- Actualmente este servidor de páginas web es compatible con los principales Sistemas de Gestión de Contenidos (CMS, del inglés *Content Management System*) del mercado tales como Wordpress, Joomla, Drupal y phpBB, por lo que puede cumplir sin problemas con las necesidades de la mayor parte de los usuarios.

### 1.3.4 Instalación y configuración

El servidor web Nginx se instala a través de la consola de Linux, en cualquier sistema operativo basado en GNU/Linux en la ruta */etc/nginx*. El archivo de configuración principal del mismo es *nginx.conf* y se puede encontrar en la siguiente ruta: */etc/nginx/nginx.conf*. En el fichero *nginx.conf*, se encuentra la información referente a la configuración general del servidor, como son: la cantidad de procesos de trabajo que utiliza, las configuraciones del tráfico HTTP, las especificaciones de las directivas del servidor de correo y las directivas que afectan al procesamiento de la conexión. Los directorios principales de configuración son *sites-available* y *sites-enabled*, además de los ficheros *conf.d*, *koi-utf mime.types*, *naxsi.rules*, *scgi_params*, *win-utf*, *fastcgi_params*, *koi-win*, *naxsi_core.rules*, *naxsi-ui.conf.1.4.1*, *proxy_params*, *uwsgi_params*. En el directorio *sites-available* se crean los archivos de configuración específicos de cada host virtual disponible, y en el directorio *sites-enabled* se crean enlaces simbólicos a los archivos de configuración que se hayan creado en *sites-available* (MOLINA, 2017).

La configuración se realiza mediante directivas, las cuales definen cómo se ejecuta Nginx. Las directivas son de dos tipos: directivas simples y directivas de bloque. Las directivas simples consisten en un conjunto de nombres y parámetros separados por espacios, y que terminan con un punto y coma. Las directivas de bloque, como su nombre indica, parece un bloque de texto delimitado por llaves {} y contiene un conjunto de directivas simples. Un archivo de configuración de Nginx típico está compuesto de bloques (SONI, 2016). Algunas directivas deben colocarse en la raíz del archivo de configuración, ya que tienen un efecto global en el servidor. La raíz del archivo de configuración también se conoce como el bloque principal (NEDELCU, 2015). Nginx se usa con frecuencia como un servidor proxy (SONI, 2016), a continuación se describe este mecanismo.

## 1.4 Nginx como proxy inverso de Apache 2

El mecanismo de proxy inverso aborda principalmente un problema: la velocidad de publicación general de Apache. Debido a la enorme cantidad de módulos y otros componentes que Apache carga en la

memoria (por cada solicitud HTTP que recibe), la PC servidora puede saturarse rápidamente cuando recibe una gran cantidad de peticiones de forma concurrente. Apache se centra en la funcionalidad a expensas de la optimización y las velocidades de procesamiento, lo que genera una memoria excesiva y una sobrecarga de CPU. Por el contrario, Nginx ha demostrado ser liviano y estable al tiempo que cumple una gran cantidad de solicitudes, utilizando menos tiempo de RAM y CPU en comparación con Apache (NEDELCU, 2015).

El objetivo principal de configurar Nginx como un servidor de interfaz y darle a Apache una función de back-end, es mejorar la velocidad de publicación. Gran cantidad de solicitudes provenientes de los clientes son para archivos estáticos y Nginx sirve mucho más rápido este tipo de archivo. El rendimiento general mejora drásticamente tanto en el lado del cliente como en el del servidor. Nginx tendría que diferenciar entre contenido estático y dinámico, en consecuencia, atender las solicitudes de archivos estáticos y reenviar las solicitudes de archivos dinámicos a un servidor back-end. Nginx actúa como un servidor proxy simple: recibe solicitudes HTTP del cliente (actuando como servidor HTTP) y las reenvía al servidor back-end (actuando como cliente HTTP) (NEDELCU, 2015).

Nginx como un servidor frontend estará en comunicación directa con el mundo exterior, mientras que Apache se ejecutará como un servidor back-end y solo intercambiará datos con Nginx. Apache puede alojarse en la misma PC servidora que el *frontend*, en cuyo caso, el puerto de escucha debe editarse para dejar el puerto 80 disponible para Nginx. Alternativamente, pueden emplearse múltiples servidores *back-end* en diferentes PC servidoras y compartir la carga. La configuración de Nginx como proxy inverso de Apache 2 tiene la ventaja que hay dos servidores web ejecutando y procesando las solicitudes, como se aprecia en la Figura 4 (NEDELCU, 2015):

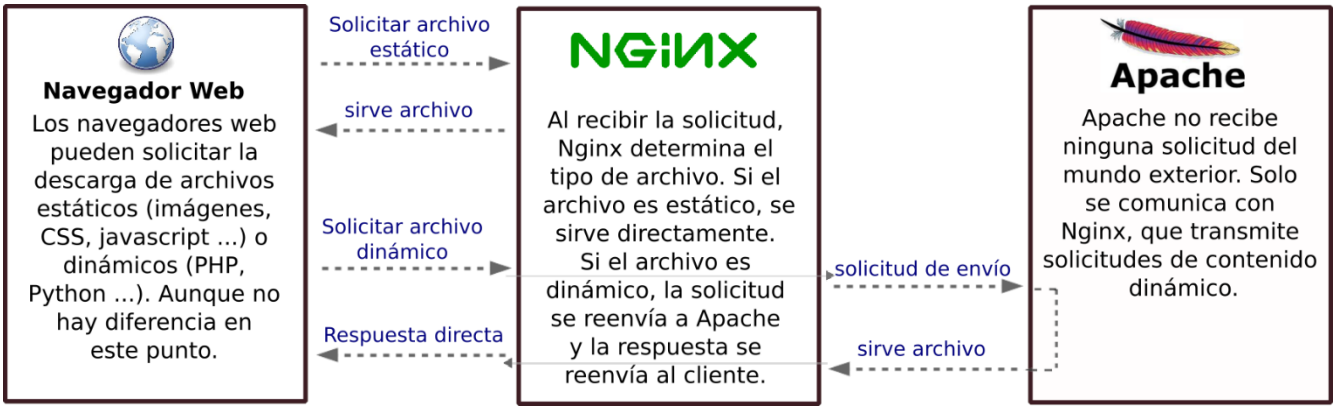


*Figura 4: Funcionamiento de Nginx como proxy inverso de Apache 2 (Fuente: elaboración propia).*

Al finalizar el presente epígrafe, se determina que la configuración de Nginx como proxy inverso de Apache 2 es una buena solución para entornos donde se desee instalar Apache 2 y a la misma vez obtener las ventajas de Nginx para el manejo de peticiones. Además de analizar cada servidor web de

forma independiente, la autora decide incluir la combinación de Nginx como proxy inverso de Apache 2 como alternativa para las instituciones cubanas. A continuación se realiza una comparación entre Apache 2 y Nginx para definir sus diferencias.

### 1.5 Comparación de Apache 2 y Nginx

De los servidores web antes mencionados se realiza una comparación haciendo referencia a las características más importantes, como se muestra en la Tabla 1.

*Tabla 1: Comparación de Apache 2 y Nginx (Fuentes: (NEDELCU, 2015) y (SONI, 2016)).*

| **Característica** | **Nginx** | **Apache 2** |
|---|---|---|
| Año de creación | 2002: Si bien Nginx es más joven que Apache, estaba destinado a una era más moderna. | 1994: Apache es uno de los numerosos proyectos de código abierto iniciados en los años 90 que contribuyeron a hacer de la World Wide Web lo que es hoy en día. |
| Lenguaje de programación en el que está desarrollado el servidor web | C: El lenguaje C es notablemente de bajo nivel y ofrece una gestión de memoria más precisa. | C y C ++: aunque Apache se escribió en C, muchos módulos se diseñaron con C ++. |
| Sistemas operativos actualmente compatibles | Multiplataforma: Nginx se ejecuta en Windows, GNU/Linux, Unix, BSD, Mac OS X y Solaris. | Multiplataforma: Apache se ejecuta en Windows, GNU/Linux, Unix, BSD, Mac OS X, Solaris, Novell NetWare, OS/2, TPF, OpenVMS, eCS, AIX, z/OS, HP-UX y más. |
| Gestión de solicitudes HTTP (cómo el servidor web procesa las solicitudes de los clientes) | Arquitectura basada en eventos: en esta arquitectura, las solicitudes se aceptan utilizando sockets asíncronos y no se procesan en hilos separados para reducir la memoria y la sobrecarga de la CPU. Los sockets asíncronos escuchan las solicitudes y los recursos se lanzan lo más rápido posible. Una vez que el cliente está listo para más datos, se desencadena otro evento y se reanuda el proceso. Inherentemente, a Nginx no le gusta bloquear sus recursos debido a la lentitud del cliente. La intención es atender tantas | Arquitectura basada en procesos y basada en hilos: en este caso, cada solicitud está en un hilo o proceso independiente y utiliza sockets síncronos. Es una arquitectura de bloqueo. Los recursos no se liberan a menos que el cliente consuma los datos. Si hay muchos clientes conectados, Apache necesitará generar más hilos para procesar las solicitudes. |

| | | |
|---|---|---|
| | solicitudes, tan rápido como sea posible. | |
| Compatibilidad con HTTPS: esto especifica si el servidor web puede brindar páginas web seguras | Soportado como un módulo: si necesita soporte HTTPS, debe asegurarse de compilar Nginx con el módulo adecuado. | Soportado como un módulo: Apache viene con soporte HTTPS a través de un módulo incluido por defecto. |
| Alojamiento virtual: esto especifica si el servidor web puede alojar múltiples sitios web en la misma computadora | Soportado de forma nativa: Nginx admite nativamente el alojamiento virtual, pero no está configurado de forma predeterminada para aceptar archivos de configuración de host virtual. | Soportado de forma nativa: Apache admite nativamente el alojamiento virtual y ofrece la posibilidad de incluir un archivo de configuración por carpeta (.htaccess). |
| Soporte para protocolos basados en CGI | FastCGI, uWSGI, SCGI: Nginx admite FastCGI, uWSGI y SCGI (así como HTTP proxying) a través de módulos que se incluyen por defecto en tiempo de compilación. | CGI, FastCGI: la mayoría de los protocolos CGI se pueden explotar a través de módulos que se pueden cargar en Apache. |
| Tipo de contenido que sirven | Nginx tiene una clara ventaja cuando se sirve contenido estático. Nginx casi siempre requiere un esfuerzo extra para que funcione con PHP, Python, Perl y otros lenguajes. A pesar de que PHP tiene un buen soporte en Nginx, aún necesita dedicar un poco de tiempo para obtener soluciones basadas en PHP que funcionen directamente en Nginx. | Apache tiene una ventaja clara y temprana para el contenido dinámico. Tiene soporte incorporado para PHP, Python, Perl y muchos otros lenguajes. Apache podría ser una mejor opción para Python o Ruby ya que no necesitará CGI para ejecutarlo. |
| Sistema de módulos: esto especifica cómo el servidor web maneja los módulos. | Sistema de módulos estático: los módulos están incorporados; deben incluirse en tiempo de compilación. | Sistema de módulos dinámico: los módulos se pueden cargar y descargar dinámicamente desde los archivos de configuración. |
| Cantidad de módulos | Nginx tiene más de 100 módulos disponibles para su uso. Pero aún hay margen de mejora. | Apache ha existido por más tiempo y tiene una ventaja definida aquí. Apache hace referencia a más de 500 módulos. |

| | | |
|---|---|---|
| Requisitos de hardware | El requisito de RAM es comparativamente menor debido a un manejo de solicitudes más eficiente. | Debido a la arquitectura, una memoria RAM menor puede crear un cuello de botella para procesar las solicitudes, ya que las solicitudes no se publican rápidamente. |
| Rendimiento | Los tiempos de respuesta son mucho más rápidos que Apache y gana sin problemas. Es uno de los principales motivos por los que las personas cambian a Nginx. | Los tiempos de respuesta son más lentos y con más solicitudes, tiende a ser aún más lento. |

El autor SONI (2016) cuando realiza la comparación de Nginx con Apache, termina con una cita famosa de Chris Lea, "*Apache es como Microsoft Word, tiene un millón de opciones, pero solo necesita seis. Nginx hace esas seis cosas, y hace cinco de ellas 50 veces más rápido que Apache*". A partir de la comparación realizada en la anterior tabla, se puede llegar a la conclusión que Apache 2 y Nginx tienen muchas características en común, sin embargo la diferencia de mayor impacto entre ambos servidores es la forma en que atienden las peticiones realizadas por el usuario. Esto proporciona quizás la diferencia más significativa en la forma en que responden a las diferentes condiciones del tráfico (CAMPOVERDE, HERNÁNDEZ y MAZÓN, 2015). En cuanto al funcionamiento del servidor web, el otro elemento que distingue a ambos servidores es el soporte para protocolos basados en CGI, la forma en que procesan el contenido dinámico. A partir del estudio realizado en el presente capítulo, la autora elaboró el esquema que se aprecia en la Figura 5.

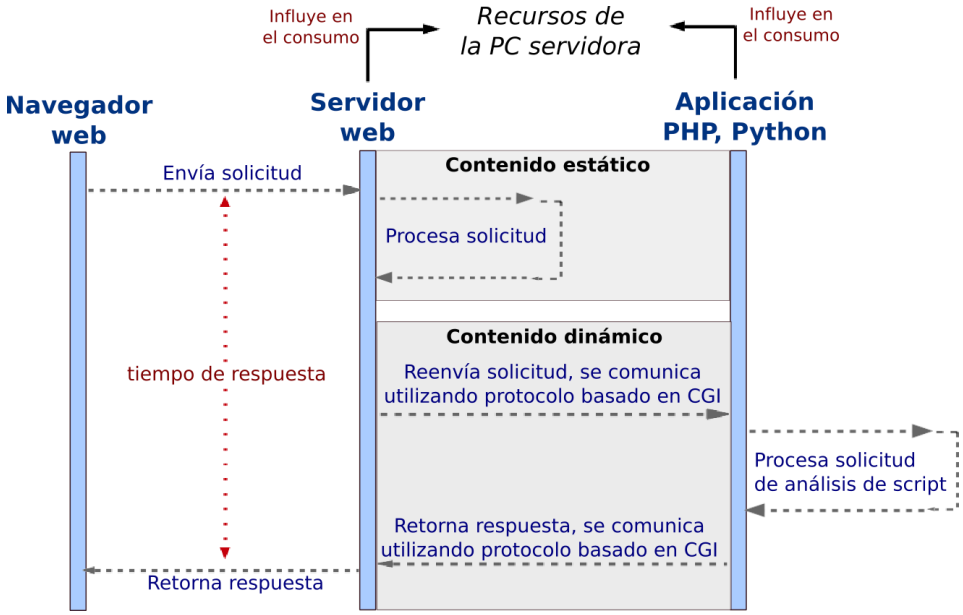


*Figura 5: Esquema de funcionamiento del servidor web (Fuente: elaboración propia).*

Como se puede observar, el tiempo de respuesta y el consumo de recursos dependen de cómo el servidor maneje las peticiones que recibe y la forma en que procese el contenido a través de los protocolos basados en CGI. Por tanto la autora determina que el correcto funcionamiento del servidor web depende de la arquitectura, la cantidad de peticiones concurrentes y el tipo de contenido publicado.

### 1.6 Conclusiones parciales

Al finalizar el presente capítulo se concluye lo siguiente:

- El análisis de las fuentes bibliográficas relacionadas con los servidores web Apache 2 y Nginx, permitió caracterizar ambos servidores e incluir como alternativa para las instituciones cubanas, la configuración de Nginx como proxy inverso de Apache 2.
- La comparación realizada entre los servidores web Apache 2 y Nginx permitió identificar que la arquitectura es la principal diferencia entre ellos.
- El estudio realizado en el presente capítulo permitió realizar un esquema del funcionamiento de un servidor web, determinando que este depende de la arquitectura del servidor, la cantidad de peticiones concurrentes y el tipo de contenido publicado.

# CAPÍTULO 2. SOLUCIÓN PARA LA SELECCIÓN DE APACHE 2 Y NGINX

En el presente capítulo se desarrolla la teoría necesaria para elaborar la solución a partir de un estudio de caso cuya finalidad es definir la diferencia entre la eficiencia de los servidores web Apache 2 y Nginx. Posteriormente se define la propuesta de solución de la investigación que se constituye como parte de la Herramienta para la Migración y Administración de Servicios Telemáticos.

## 2.1 Estudio de caso para determinar la eficiencia de Apache 2 y Nginx

La eficiencia en la selección de los servidores web Apache 2 y Nginx implica el desempeño adecuado del servidor web que se seleccione para un entorno determinado. Por tanto, se hace necesario el estudio y análisis de la eficiencia de ambos servidores en un entorno de prueba, por lo que se realizará un estudio de caso. La investigación de estudios de casos se usa ampliamente en muchas disciplinas (MCWHORTER y ELLINGER, 2018), pueden ser: descriptivas, si lo que se pretende es identificar y describir los distintos factores que ejercen influencia en el fenómeno estudiado, y exploratorias, si a través de las mismas se pretende conseguir un acercamiento entre las teorías inscritas en el marco teórico y la realidad del objeto de estudio (MARTÍNEZ, 2006).

Los estudios de casos se refieren a estudios que al utilizar los procesos de investigación cuantitativa, cualitativa o mixta, analizan profundamente una unidad para responder al planteamiento del problema, probar hipótesis y desarrollar alguna teoría. En ocasiones utilizan la experimentación, se constituyen en estudios preexperimentales. La unidad o caso investigado puede tratarse de un individuo, una familia, un objeto, un sistema, una organización, etc. (HERNÁNDEZ, FERNÁNDEZ y BAPTISTA, 2006). A continuación se definen elementos que caracterizan el estudio de caso a realizar.

**Objetivo del estudio:** definir la diferencia entre los servidores web Apache 2 y Nginx en cuanto a la eficiencia de desempeño, teniendo en cuenta los indicadores especificados en el Anexo 3.

**Unidad de análisis:** eficiencia de Apache 2 y Nginx.

**Descripción:** se realiza un estudio preexperimental, que consiste en administrar un tratamiento a un grupo y después aplicar una medición de una o más variables, para observar cuál es el nivel del grupo en estas variables (HERNÁNDEZ, FERNÁNDEZ y BAPTISTA, 2006). Para la medición de la eficiencia se emplean los indicadores de la Norma Cubana ISO/IEC 25023:2017 (ver Anexo 3), esta es una adopción idéntica por el método de traducción de la Norma Internacional *Systems and software engineering - Systems and software Quality Requirements and Evaluation* (SQuaRE). Esta Norma Internacional define las medidas de la calidad para evaluar cuantitativamente la calidad del producto de software y del sistema en términos de características y subcaracterísticas. La Norma Cubana ISO/IEC 25023:2017 fue elaborada por la Oficina Nacional de Normalización (NC), que es el Organismo

Nacional de Normalización de la República de Cuba y representa al país ante las organizaciones internacionales y regionales de normalización (OFICINA NACIONAL DE NORMALIZACIÓN, 2017).

La PC servidora se estudiará en tres escenarios de prueba diferentes en cuanto al tipo de contenido que puede ser: estático, dinámico con PHP o dinámico con Python. A su vez para el tipo de contenido estático existen dos escenarios en cuanto al servidor web instalado que puede ser Apache 2 o Nginx, por otra parte para tipo de contenido dinámico con ambos lenguajes de programación se encuentran tres escenarios: Apache 2, Nginx o Nginx funcionando como proxy inverso de Apache 2 (la autora de la investigación lo denominó Proxy). De lo anteriormente explicado se concluye que para cada uno de los 9 indicadores de la variable eficiencia de desempeño, se estudiarán ocho escenarios.

El estudio consistió en realizar con la herramienta *ab* (*Apache Benchmark*) un número determinado de peticiones con cierta concurrencia, desde una PC cliente al servidor web instalado en la PC servidora. En cada uno de los ocho escenarios se realizaron 10 observaciones, para todas con un total de 50 000 peticiones y cada una con concurrencias de 10, 100, 250, 500, 750, 1 000, 5 000, 10 000, 15 000 y 20 000 peticiones respectivamente. Además al mismo tiempo se realizó el monitoreo de los recursos de la PC servidora con la herramienta *dstat*. Las características de ambas PC se aprecian en la Tabla 2.

*Tabla 2: Características de las dos PC empleadas en el estudio de caso (Fuente: elaboración propia).*

| Identificador | RAM | CPU | HDD | Sistema Operativo |
|---|---|---|---|---|
| PC-Servidora | 8 GB | Intel Core i3-2120 de 3.30 GHz y 4 procesadores | 1024 GB | Nova Servidores 6.0 |
| PC-Cliente | 4 GB | Intel Core i3-2120 de 3.30 GHz y 4 procesadores | 1024 GB | Nova Escritorio 6.0 |

**Herramientas informáticas empleadas**

Para la ejecución del proceso anteriormente descrito se emplearon las herramientas *ab* y *dstat*.

- *ab* es una herramienta para la evaluación comparativa del servidor web HTTP (KUNDA, CHIHANA y SINYINDA, 2017). Está diseñada para dar una impresión de cómo funciona el servidor web, mostrando especialmente cuántas solicitudes por segundo puede servir. Mediante la misma se realizan un total de peticiones *n*, con una concurrencia *c* (donde el valor máximo que acepta es 20 000) a una URL determinada.
- *dstat* es una herramienta versátil para generar estadísticas de recursos del sistema, es un reemplazo versátil para iostat, ifstat y vmstat. *Dstat* supera algunas de las limitaciones y agrega algunas características adicionales. Permite ver todos los recursos del sistema al instante, también proporciona inteligentemente la información más detallada en columnas e indica claramente en qué magnitud y unidad se muestra la salida. *Dstat* permite que los datos se escriban directamente en un archivo CSV para ser importados y utilizados por OpenOffice,

Gnumeric o Excel para crear gráficos. *Dstat* combina las funciones de varias herramientas en un único paquete (HÖBEL, 2013).

Para el cálculo de la eficiencia en cada indicador se definieron cinco escenarios en cuanto a la cantidad de peticiones concurrentes que pueden existir en las instituciones (denominada con la variable *p*), teniendo en cuenta las respuestas de los administradores de servidores web en la entrevista realizada, la cual se aprecia en el Anexo 4. A continuación se describen cada uno de los escenarios.

1) **p≤500**: se refiere a una cantidad de peticiones concurrentes menor o igual que 500. Se incluyen cuatro observaciones con concurrencias 10, 100, 250 y 500 respectivamente.
2) **500<p≤1000**: se refiere a una cantidad de peticiones concurrentes mayor que 500 y menor o igual que 1 000. Se incluyen dos observaciones con concurrencias 750 y 1 000 respectivamente.
3) **1000<p≤10000**: se refiere a una cantidad de peticiones concurrentes mayor que 1 000 y menor o igual que 10 000. Se incluyen dos observaciones con concurrencias 5 000 y 10 000 respectivamente.
4) **p>10000**: se refiere a una cantidad de peticiones concurrentes mayor que 10 000. Se incluyen dos observaciones con concurrencias 15 000 y 20 000 respectivamente.
5) **Todas**: se refiere a todas las concurrencias de peticiones incluyendo las 10 observaciones.

Las fórmulas empleadas para obtener la eficiencia en cada caso aparecen en el Anexo 3. Los gráficos correspondientes a cada indicador aparecen en el Anexo 4, donde el tipo de contenido estático se expresa con columnas agrupadas, el dinámico con PHP mediante líneas y el dinámico con Python a través de líneas con marcadores. A continuación se describen los resultados del estudio.

### 2.1.1 Tiempo medio de conclusión de un trabajo

El tiempo medio de conclusión de un trabajo se calcula como la sumatoria del tiempo que demora el servidor web en dar respuesta a una petición en cada observación, dividido entre la cantidad de observaciones. Este elemento se determinó con la salida de la herramienta *ab* para el parámetro *Time per request*, que expresa el tiempo en milisegundos (ms). En la Figura 12 del Anexo 5 se muestra la gráfica con los tiempos de respuesta en cada servidor web, para cada una de las 10 observaciones. Apache 2 y Nginx se demoran más tiempo para responder cuando es tipo de contenido dinámico, que para estático. Para contenido estático y dinámico con PHP, en todos los servidores se mantiene el tiempo de respuesta estable y aumenta el valor en las últimas observaciones. Para contenido dinámico con Python el tiempo de respuesta para los tres servidores es notablemente mucho mayor que para los restantes tipos de contenido. El cálculo de la eficiencia para este indicador se muestra en la Tabla 3, donde para *Todas* las observaciones con contenido estático el servidor web más eficiente es Nginx y

para ambos casos de contenido dinámico es Proxy. Para cada escenario de concurrencia aparece en negrita el valor resultante del servidor más eficiente, un valor menor es mejor.

*Tabla 3: Eficiencia del tiempo medio de conclusión de un trabajo (Fuente: elaboración propia).*

| Cantidad de Peticiones Concurrentes (p) | Tipo de contenido | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Estático | | Dinámico | | | | | |
| | | | PHP | | | Python | | |
| | Apache | Nginx | Apache | Nginx | Proxy | Apache | Nginx | Proxy |
| **p≤500** | 0,212 | **0,210** | 0,604 | **0,597** | 0,601 | 4,482 | **3,937** | 3,987 |
| **500<p≤ 1000** | 0,217 | **0,214** | 0,637 | **0,599** | 0,610 | 6,167 | 5,732 | **5,571** |
| **1000<p≤10000** | **0,460** | 0,494 | 0,871 | 0,760 | **0,643** | 6,719 | 6,180 | **5,932** |
| **p>10000** | 0,817 | **0,653** | 2,049 | 1,688 | **1,091** | 7,756 | 7,211 | **6,464** |
| **Todas** | 0,383 | **0,356** | 0,953 | 0,848 | **0,709** | 5,921 | 5,400 | **5,188** |

### 2.1.2 Adecuación del tiempo de conclusión de un trabajo

La adecuación del tiempo de conclusión de un trabajo se refiere a qué tan bien el tiempo de respuesta cumple los objetivos especificados. Se calcula como el cociente entre el tiempo medio de conclusión de un trabajo medido por el indicador anterior y el tiempo de respuesta especificado, que es 200 ms. El cálculo de la eficiencia para este indicador se muestra en la Tabla 4, donde para *Todas* las observaciones con contenido estático el servidor web más eficiente es Nginx y para ambos casos de contenido dinámico es Proxy. Para los tres tipos de contenido los servidores web disminuyen su eficiencia con el aumento de la concurrencia de peticiones. Para cada escenario de concurrencia, se muestra en negrita el valor resultante del servidor más eficiente, un valor menor es mejor.

*Tabla 4: Eficiencia de la adecuación del tiempo de conclusión de un trabajo (Fuente: elaboración propia).*

| Cantidad de Peticiones Concurrentes (p) | Tipo de contenido | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Estático | | Dinámico | | | | | |
| | | | PHP | | | Python | | |
| | Apache | Nginx | Apache | Nginx | Proxy | Apache | Nginx | Proxy |
| **p≤500** | 0,00106 | **0,00105** | 0,00302 | **0,00298** | 0,00300 | 0,02241 | **0,01969** | 0,01994 |
| **500<p≤ 1000** | 0,00109 | **0,00107** | 0,00319 | **0,00299** | 0,00305 | 0,03083 | 0,02866 | **0,02785** |
| **1000<p≤10000** | **0,00230** | 0,00247 | 0,00435 | 0,00380 | **0,00322** | 0,03359 | 0,03090 | **0,02966** |
| **p>10000** | 0,00409 | **0,00326** | 0,01024 | 0,00844 | **0,00545** | 0,03878 | 0,03606 | **0,03232** |
| **Todas** | 0,00192 | **0,00178** | 0,00476 | 0,00424 | **0,00354** | 0,02961 | 0,02700 | **0,02594** |

### 2.1.3 Rendimiento medio

El rendimiento medio se refiere a la cantidad de peticiones que el servidor web puede atender en un segundo. Se calcula como la sumatoria del cociente entre el número de peticiones y el tiempo de la observación, dividido por el número de observaciones. El número de peticiones se determinó con la salida de la herramienta *ab* para el parámetro *Completed requests* (cuando se completaron todas las peticiones) o *Total of requests completed* (cuando no se completaron todas). El tiempo de la observación se determinó con la misma herramienta para el parámetro *Time taken for tests*, que expresa el tiempo en segundos (s). En la Figura 13 del Anexo 5 se muestra la gráfica con la cantidad de peticiones completadas por segundo, para cada una de las 10 observaciones. Para contenido estático los dos servidores web poseen un mayor rendimiento que para contenido dinámico, sin embargo en las últimas observaciones disminuyen la cantidad de peticiones completadas. Para contenido dinámico los tres servidores web poseen un mayor rendimiento con PHP con respecto a Python, en las últimas observaciones la cantidad de peticiones completadas decrece. El cálculo de la eficiencia para este indicador se muestra en la Tabla 5, donde para *Todas* las observaciones con contenido estático el servidor web más eficiente es Nginx y para ambos casos de contenido dinámico es Proxy. Para cada escenario de concurrencia aparece en negrita el valor resultante del servidor más eficiente, un valor mayor es mejor.

*Tabla 5: Eficiencia del rendimiento medio (Fuente: elaboración propia).*

| Cantidad de Peticiones Concurrentes (p) | Tipo de contenido | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Estático | | Dinámico | | | | | |
| | | | PHP | | | Python | | |
| | Apache | Nginx | Apache | Nginx | Proxy | Apache | Nginx | Proxy |
| **p≤500** | 4729,3 | **4764,6** | 1655,5 | **1676,0** | 1665,1 | 251,2 | **291,0** | 288,0 |
| **500<p≤ 1000** | 4607,7 | **4688,8** | 1569,4 | **1671,7** | 1640,5 | 162,3 | 174,6 | **179,6** |
| **1000<p≤10000** | 2813,0 | **2894,8** | 344,0 | 1377,9 | **1560,5** | 148,9 | 161,8 | **168,6** |
| **p>10000** | 2219,3 | **2846,0** | 315,5 | 596,2 | **1136,8** | 129,5 | 139,9 | **154,8** |
| **Todas** | 3819,7 | **3991,8** | 1108,0 | 1399,5 | **1533,6** | 188,6 | 211,7 | **215,8** |

### 2.1.4 La media de utilización del procesador

La media de utilización del procesador se refiere al tiempo del procesador que se utiliza para ejecutar un determinado conjunto de tareas en comparación con el tiempo de operación. Se calcula como la sumatoria del cociente entre el tiempo del procesador realmente utilizado (determinado por la opción *--top-cputime-avg* de *dstat*) y el tiempo de operación (opción *Time taken for tests* de *ab*), dividido por el número de observaciones. En la Figura 14 del Anexo 5 se muestra la gráfica con el tiempo del procesador realmente utilizado por cada servidor web, para cada una de las 10 observaciones. Para

contenido estático los dos servidores web utilizan menos tiempo del procesador que para contenido dinámico. Nginx para todas las observaciones se mantiene como el servidor más eficiente, utilizando menor cantidad de tiempo que Apache. Para contenido dinámico con PHP y Python, Apache 2 utiliza considerablemente más tiempo que los demás servidores. El cálculo de la eficiencia para este indicador se muestra en la Tabla 6, donde para *Todas* las observaciones en cada tipo de contenido el servidor web más eficiente es Nginx. Para cada escenario de concurrencia aparece en negrita el valor resultante del servidor más eficiente, un valor menor es mejor.

*Tabla 6: Eficiencia de la media de utilización del procesador (Fuente: elaboración propia).*

| Cantidad de Peticiones Concurrentes (p) | Tipo de contenido | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Estático | | Dinámico | | | | | |
| | | | PHP | | | Python | | |
| | Apache | Nginx | Apache | Nginx | Proxy | Apache | Nginx | Proxy |
| **p≤500** | 0,0649 | **0,0195** | 0,0325 | **0,0072** | 0,0092 | 0,0044 | **0,0031** | 0,0032 |
| **500<p≤ 1000** | 0,0559 | **0,0061** | 0,0408 | 0,0084 | **0,0048** | 0,0048 | **0,0013** | 0,0020 |
| **1000<p≤10000** | 0,0375 | **0,0116** | 0,0338 | **0,0076** | 0,0168 | 0,0048 | **0,0020** | 0,0026 |
| **p>10000** | 0,0349 | **0,0199** | 0,5057 | **0,0071** | 0,0221 | 0,0050 | **0,0027** | 0,0032 |
| **Todas** | 0,0516 | **0,0153** | 0,1291 | **0,0075** | 0,0124 | 0,0047 | **0,0024** | 0,0028 |

### 2.1.5 La media de utilización de la memoria

En cuanto a la utilización de la memoria, durante la ejecución de la prueba no se usó la swap, la capacidad de consumo reportada fue de 0 MB tanto para contenido estático como dinámico con PHP y Python. Este indicador se calcula como la sumatoria del cociente entre el consumo de RAM (se obtuvo a través de la opción *–m* de *dstat*, específicamente la columna para la memoria usada) y la memoria disponible que es la capacidad total de RAM de la PC servidora (8192 MB), dividido por el número de observaciones. Para obtener el consumo de memoria en cada observación, se calculó la media de todos los números durante el tiempo de la prueba, obteniendo un valor promedio para cada servidor en dependencia del tipo de contenido y de la cantidad de peticiones concurrentes, ver Figura 15 del Anexo 5, donde se aprecian cada una de las 10 observaciones. Para contenido estático, los dos servidores web consumen menos RAM que para contenido dinámico. Apache 2 es el servidor que más memoria utiliza, además el consumo aumenta más a mayor concurrencia de peticiones. Para contenido dinámico los tres servidores web utilizan más memoria con Python que con PHP. En ambos casos Apache utiliza notablemente más memoria que los demás servidores. El cálculo de la eficiencia para este indicador se muestra en la Tabla 7, donde para *Todas* las observaciones con cada tipo de contenido el servidor web más eficiente es Nginx. Para cada escenario de concurrencia aparece en negrita el valor resultante del servidor más eficiente, un valor menor es mejor.

*Tabla 7: Eficiencia de la media de utilización de la memoria (Fuente: elaboración propia).*

| Cantidad de Peticiones Concurrentes (p) | Tipo de contenido | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Estático | | Dinámico | | | | | |
| | | | PHP | | | Python | | |
| | Apache | Nginx | Apache | Nginx | Proxy | Apache | Nginx | Proxy |
| **p≤500** | 0,059 | **0,034** | 0,083 | **0,036** | 0,042 | 0,090 | **0,040** | 0,056 |
| **500<p≤ 1000** | 0,061 | **0,035** | 0,144 | **0,038** | 0,047 | 0,157 | **0,041** | 0,061 |
| **1000<p≤10000** | 0,097 | **0,043** | 0,146 | **0,055** | 0,066 | 0,160 | **0,056** | 0,081 |
| **p>10000** | 0,149 | **0,051** | 0,135 | **0,074** | 0,116 | 0,150 | **0,077** | 0,131 |
| **Todas** | 0,085 | **0,039** | 0,118 | **0,048** | 0,063 | 0,129 | **0,051** | 0,077 |

### 2.1.6 La media del uso de los dispositivos de entrada/salida (E/S)

La media del uso de los dispositivos de entrada/salida se refiere al período de tiempo durante el cual un sistema o un dispositivo están realmente funcionando, en este caso el disco duro. Se calcula como la sumatoria del cociente entre el tiempo ocupado de los dispositivos y la duración de las operaciones de E/S, dividido entre la cantidad de observaciones. Se calculó a partir de la opción *--disk-util* de *dstat*, que devuelve la utilización del disco en porciento. Aplicando la regla de tres se sustituyó en la fórmula el cociente entre el porciento y 100, en lugar del cociente entre el tiempo ocupado de los dispositivos y la duración de las operaciones de E/S. En la Figura 16 del Anexo 5 se muestra la gráfica con el uso de los dispositivos de entrada salida en porciento, para cada una de las 10 observaciones. En los tres tipos de contenido el valor para cada servidor web se mantiene variable. El cálculo de la eficiencia para este indicador se muestra en la Tabla 8, donde para *Todas* las observaciones en cada tipo de contenido el servidor web más eficiente es Nginx. Para cada escenario de concurrencia aparece en negrita el valor resultante del servidor más eficiente, un valor menor es mejor.

*Tabla 8: Eficiencia del uso de los dispositivos de E/S (Fuente: elaboración propia).*

| Cantidad de Peticiones Concurrentes (p) | Tipo de contenido | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Estático | | Dinámico | | | | | |
| | | | PHP | | | Python | | |
| | Apache | Nginx | Apache | Nginx | Proxy | Apache | Nginx | Proxy |
| **p≤500** | **0,01633** | 0,02033 | 0,01503 | **0,01490** | 0,01633 | 0,01733 | 0,01798 | **0,01728** |
| **500<p≤ 1000** | **0,01600** | 0,02000 | **0,01370** | 0,01435 | 0,01880 | 0,01835 | **0,00865** | 0,01915 |
| **1000<p≤10000** | 0,02250 | **0,01150** | 0,01840 | 0,01600 | **0,01340** | 0,01965 | **0,01545** | 0,02120 |
| **p>10000** | 0,01870 | **0,01500** | 0,02070 | **0,01770** | 0,02230 | 0,02050 | **0,01800** | 0,02110 |
| **Todas** | 0,01797 | **0,01743** | 0,01657 | **0,01557** | 0,01743 | 0,01863 | **0,01561** | 0,01920 |

### 2.1.7 Utilización del ancho de banda

La utilización del ancho de banda se refiere a qué proporción del ancho de banda disponible se utiliza para realizar un conjunto dado de tareas. Se calcula como el cociente entre el ancho de banda de transmisión real medido a lo largo del tiempo (con la opción *–n* de *dstat*) y la capacidad de ancho de banda disponible (10 Mbps). En la Figura 17 del Anexo 5 se muestra la gráfica con el ancho de banda utilizado medido en Mbps, para cada una de las 10 observaciones. Para contenido estático se utiliza más ancho de banda que para contenido dinámico, solo decrece en las últimas observaciones. Los tres servidores web realizan menor uso del ancho de banda para contenido dinámico con Python que cuando es con PHP. En la mayoría de las observaciones Nginx realiza menor consumo que los restantes servidores web. El cálculo de la eficiencia para este indicador se muestra en la Tabla 9, donde para *Todas* las observaciones en cada tipo de contenido el servidor web más eficiente es Nginx. Para cada escenario de concurrencia aparece en negrita el valor resultante del servidor más eficiente, un valor menor es mejor.

*Tabla 9: Eficiencia de la utilización del ancho de banda (Fuente: elaboración propia).*

| Cantidad de Peticiones Concurrentes (p) | Tipo de contenido | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Estático | | Dinámico | | | | | |
| | | | PHP | | | Python | | |
| | Apache | Nginx | Apache | Nginx | Proxy | Apache | Nginx | Proxy |
| **p≤500** | 2,834 | **2,742** | 1,748 | **1,619** | 1,660 | 0,211 | **0,208** | 0,218 |
| **500<p≤ 1000** | 2,764 | **2,695** | 1,665 | **1,411** | 1,624 | 0,131 | **0,074** | 0,134 |
| **1000<p≤10000** | 1,763 | **1,603** | 1,547 | **1,262** | 1,434 | 0,140 | **0,067** | 0,116 |
| **p>10000** | **1,845** | 1,940 | 1,476 | **1,195** | 1,468 | 0,153 | **0,070** | 0,118 |
| **Todas** | 2,408 | **2,345** | 1,637 | **1,421** | 1,569 | 0,170 | **0,125** | 0,161 |

### 2.1.8 Capacidad de procesamiento de transacciones

La capacidad de procesamiento de transacciones se refiere a las transferencias realizadas en un segundo. Se calcula como el cociente entre la cantidad de transacciones y el tiempo de la observación (opción *Time taken for tests* de *ab*). La cantidad de transacciones se obtuvo con la salida de la herramienta *dstat* para la opción *--disk-tps*, que devuelve las transacciones de disco por segundo (tps). En la Figura 18 del Anexo 5, se muestra la gráfica con las tps por cada servidor web, para cada una de las 10 observaciones. En los tres tipos de contenido el valor para cada servidor web se mantiene variable. El cálculo de la eficiencia para este indicador se muestra en la Tabla 10, donde para *Todas* las observaciones con contenido estático el servidor web más eficiente es Apache 2 y para contenido

dinámico con ambos lenguajes de programación es Nginx. Para cada escenario de concurrencia aparece en negrita el valor resultante del servidor más eficiente, un valor menor es mejor.

*Tabla 10: Eficiencia de la capacidad de procesamiento de transacciones (Fuente: elaboración propia).*

| Cantidad de Peticiones Concurrentes (p) | Tipo de contenido | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Estático | | Dinámico | | | | | |
| | | | PHP | | | Python | | |
| | Apache | Nginx | Apache | Nginx | Proxy | Apache | Nginx | Proxy |
| **p≤500** | **1,65** | 2,92 | 2,63 | **1,67** | 3,35 | 2,55 | 2,18 | **2,16** |
| **500<p≤ 1000** | **1,33** | 1,67 | 2,43 | **1,88** | 2,19 | 2,74 | **2,00** | 2,71 |
| **1000<p≤10000** | 2,56 | **1,35** | 3,14 | 2,42 | **1,55** | 3,16 | **2,00** | 2,86 |
| **p>10000** | **2,60** | 4,19 | 3,09 | **2,70** | 4,45 | 3,40 | **2,53** | 3,11 |
| **Todas** | **1,96** | 2,61 | 2,78 | **2,06** | 2,98 | 2,92 | **2,18** | 2,60 |

### 2.1.9 Capacidad de acceso de usuario

La capacidad de acceso de usuario se refiere a la capacidad máxima de usuarios que acceden en un momento determinado. La eficiencia se calcula como la sumatoria de la cantidad de usuarios que acceden en cada observación, dividida entre el número de observaciones. En la Figura 19 del Anexo 5 se muestra la gráfica con los valores en cada servidor web, para cada una de las 10 observaciones en los tres tipos de contenido. Para contenido estático, Apache 2 y Nginx presentan mayor capacidad de acceso de usuario que para contenido dinámico. Para contenido dinámico con ambos lenguajes de programación los tres servidores web disminuyen la capacidad de acceso de usuario en las últimas observaciones, sin embrago con Python presentan menor capacidad que con PHP. El cálculo de la eficiencia para este indicador se muestra en la Tabla 11, donde para *Todas* las observaciones con contenido estático el servidor más eficiente es Nginx y para contenido dinámico con ambos lenguajes de programación es Proxy. Para cada escenario de concurrencia aparece en negrita el valor resultante del servidor más eficiente, un valor mayor es mejor, en este caso existen escenarios con más de un servidor con la misma eficiencia.

*Tabla 11: Eficiencia para la capacidad de acceso de usuario (Fuente: elaboración propia).*

| Cantidad de Peticiones Concurrentes (p) | Tipo de contenido | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Estático | | Dinámico | | | | | |
| | | | PHP | | | Python | | |
| | Apache | Nginx | Apache | Nginx | Proxy | Apache | Nginx | Proxy |
| **p≤500** | **50000,0** | **50000,0** | **50000,0** | **50000,0** | **50000,0** | 49369,3 | 49438,3 | **50000,0** |
| **500<p≤ 1000** | **50000,0** | **50000,0** | **50000,0** | **50000,0** | **50000,0** | 47907,0 | 48481,0 | **50000,0** |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **1000<p≤10000** | **50000,0** | **50000,0** | 45138,5 | **50000,0** | **50000,0** | 43817,0 | 49847,5 | **49938,5** |
| **p>10000** | 49998,5 | **50000,0** | 37044,0 | 38832,5 | **49984,5** | 36411,5 | 46890,0 | **47879,0** |
| **Todas** | 49999,7 | **50000,0** | 46436,5 | 47766,5 | **49996,9** | 45374,8 | 48819,0 | **49563,5** |

### 2.1.10 Resultado del análisis

A partir del análisis de la eficiencia calculada en los epígrafes anteriores, con el objetivo de medir todos los indicadores en una misma escala, se asignó en cada escenario en cuanto a la concurrencia de peticiones y atendiendo al tipo de contenido, los valores 3, 2 y 1 a los servidores web en correspondencia con el orden de su eficiencia, de mayor a menor. Para casos de igual eficiencia se asignó el mismo valor. Para cada uno de los cinco escenarios en cuanto a la concurrencia de peticiones, se calculó la puntuación final para cada servidor web teniendo en cuenta todos los indicadores y además para cada conjunto de indicadores correspondientes a las tres medidas de la eficiencia que son Rendimiento, Utilización de los recursos y Capacidad, como se aprecia en la Tabla 12.

*Tabla 12: Resultado del análisis de los indicadores (Fuente: elaboración propia).*

| **Cantidad de Peticiones Concurrentes (p)** | **Indicador** | **Tipo de contenido** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Estático** | | **Dinámico** | | | | | |
| | | | | **PHP** | | | **Python** | | |
| | | **Apache** | **Nginx** | **Apache** | **Nginx** | **Proxy** | **Apache** | **Nginx** | **Proxy** |
| **p≤500** | Rendimiento | | | | | | | | |
| | **1** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **2** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **3** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | | **6** | **9** | **3** | **9** | **6** | **3** | **9** | **6** |
| | Utilización de los recursos | | | | | | | | |
| | **4** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **5** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **6** | 3 | 2 | 2 | 3 | 1 | 2 | 1 | 3 |
| | **7** | 2 | 3 | 1 | 3 | 2 | 2 | 3 | 1 |
| | | **9** | **11** | **5** | **12** | **7** | **6** | **10** | **8** |
| | Capacidad | | | | | | | | |
| | **8** | 3 | 2 | 2 | 3 | 1 | 1 | 2 | 3 |
| | **9** | 3 | 3 | 3 | 3 | 3 | 1 | 2 | 3 |
| | | **6** | **5** | **5** | **6** | **4** | **2** | **4** | **6** |
| **Puntuación final** | | **21** | **25** | **13** | **27** | **17** | **11** | **23** | **20** |
| **500<p≤1000** | Rendimiento | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **1** | 2 | 3 | 1 | 3 | 2 | 1 | 2 | 3 |
| | **2** | 2 | 3 | 1 | 3 | 2 | 1 | 2 | 3 |
| | **3** | 2 | 3 | 1 | 3 | 2 | 1 | 2 | 3 |
| | | **6** | **9** | **3** | **9** | **6** | **3** | **6** | **9** |
| | Utilización de los recursos | | | | | | | | |
| | **4** | 2 | 3 | 1 | 2 | 3 | 1 | 3 | 2 |
| | **5** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **6** | 3 | 2 | 3 | 2 | 1 | 2 | 3 | 1 |
| | **7** | 2 | 3 | 1 | 3 | 2 | 2 | 3 | 1 |
| | | **9** | **11** | **6** | **10** | **8** | **6** | **12** | **6** |
| | Capacidad | | | | | | | | |
| | **8** | 3 | 2 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **9** | 3 | 3 | 3 | 3 | 3 | 1 | 2 | 3 |
| | | **6** | **5** | **4** | **6** | **5** | **2** | **5** | **5** |
| **Puntuación final** | | **21** | **25** | **13** | **25** | **19** | **11** | **23** | **20** |
| **1000<p≤10000** | Rendimiento | | | | | | | | |
| | **1** | 3 | 2 | 1 | 2 | 3 | 1 | 2 | 3 |
| | **2** | 3 | 2 | 1 | 2 | 3 | 1 | 2 | 3 |
| | **3** | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| | | **8** | **7** | **3** | **6** | **9** | **3** | **6** | **9** |
| | Utilización de los recursos | | | | | | | | |
| | **4** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **5** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **6** | 2 | 3 | 1 | 2 | 3 | 2 | 3 | 1 |
| | **7** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | | **8** | **12** | **4** | **11** | **9** | **5** | **12** | **7** |
| | Capacidad | | | | | | | | |
| | **8** | 2 | 3 | 1 | 2 | 3 | 1 | 3 | 2 |
| | **9** | 3 | 3 | 2 | 3 | 3 | 1 | 2 | 3 |
| | | **5** | **6** | **3** | **5** | **6** | **2** | **5** | **5** |
| **Puntuación final** | | **21** | **25** | **10** | **22** | **24** | **10** | **23** | **21** |
| **p>10000** | Rendimiento | | | | | | | | |
| | **1** | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| | **2** | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| | **3** | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| | | **6** | **9** | **3** | **6** | **9** | **3** | **6** | **9** |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Utilización de los recursos | | | | | | | | |
| | **4** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **5** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **6** | 2 | 3 | 2 | 3 | 1 | 2 | 3 | 1 |
| | **7** | 3 | 2 | 1 | 3 | 2 | 1 | 3 | 2 |
| | | **9** | **11** | **5** | **12** | **7** | **5** | **12** | **7** |
| | Capacidad | | | | | | | | |
| | **8** | 3 | 2 | 2 | 3 | 1 | 1 | 3 | 2 |
| | **9** | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| | | **5** | **5** | **3** | **5** | **4** | **2** | **5** | **5** |
| **Puntuación final** | | **20** | **25** | **11** | **23** | **20** | **10** | **23** | **21** |
| **Todas** | Rendimiento | | | | | | | | |
| | **1** | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| | **2** | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| | **3** | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| | | **6** | **9** | **3** | **6** | **9** | **3** | **6** | **9** |
| | Utilización de los recursos | | | | | | | | |
| | **4** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **5** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | **6** | 2 | 3 | 2 | 3 | 1 | 2 | 3 | 1 |
| | **7** | 2 | 3 | 1 | 3 | 2 | 1 | 3 | 2 |
| | | **8** | **12** | **5** | **12** | **7** | **5** | **12** | **7** |
| | Capacidad | | | | | | | | |
| | **8** | 3 | 2 | 2 | 3 | 1 | 1 | 3 | 2 |
| | **9** | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| | | **5** | **5** | **3** | **5** | **4** | **2** | **5** | **5** |
| **Puntuación final** | | **19** | **26** | **11** | **23** | **20** | **10** | **23** | **21** |

La elaboración de la tabla anterior como resultado de la aplicación del estudio de caso, permitió realizar la puntuación de cada servidor web e identificar la diferencia entre los mismos en cuanto a la eficiencia en sentido general y respecto a sus tres medidas: Rendimiento, Utilización de los recursos y Capacidad. Partiendo de la teoría definida, a continuación se desarrollará la solución como parte de la Herramienta para la Migración y Administración de Servicios Telemáticos.

## 2.2 Herramienta para la Migración y Administración de Servicios Telemáticos

La Herramienta para la Migración y Administración de Servicios Telemáticos (HMAST) en su versión 2.0 contiene varios módulos, dos de ellos corresponden a los servidores web Apache 2 y Nginx respectivamente. La herramienta permite la administración remota a través del protocolo SSH (*Secure SHell*) de diferentes PC servidoras, como se muestra en la Figura 6. La solución informática de la presente investigación se sustenta en añadir a HMAST un componente Web que permita seleccionar el servidor web que se ajuste a la institución, a partir del resultado obtenido con la aplicación del estudio de caso del epígrafe anterior.

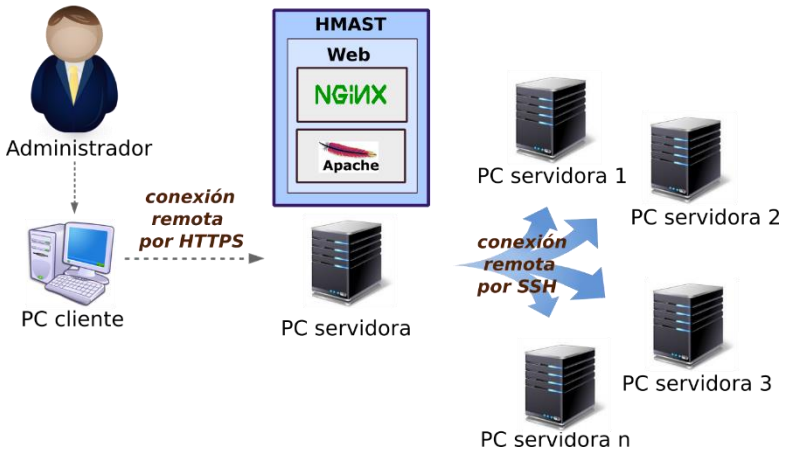


*Figura 6: Esquema de funcionamiento de HMAST (Fuente: elaboración propia).*

La instalación de la herramienta en una PC diferente a la que está instalado el servidor, evita el consumo adicional de recursos. La característica de modularidad de HMAST permite agregar más módulos, sin que esto afecte a los restantes, en caso que en las instituciones se necesite el empleo de otro servidor web. Existen dos funcionalidades que se aplican a todos los módulos, las cuales se describen a continuación.

**Aplicar cambios al servidor:** el sistema permite guardar las configuraciones locales del servicio en la PC servidora. El administrador del sistema o del módulo debe seleccionar la opción Aplicar cambios. Al ejecutar la opción se verifica que las configuraciones de la PC servidora no hayan sufrido cambios (MOLINA, 2016).

- Si las configuraciones de la PC servidora no han sufrido cambios se le muestra al usuario el mensaje: ¿Está seguro que desea salvar toda la configuración en el servidor? Esto reconfigurará el servicio con los datos establecidos. Luego el usuario debe seleccionar la opción Aceptar para guardar los cambios o la opción Cancelar para mantener la configuración anterior. Si selecciona la opción Aceptar, se muestra el mensaje: Los cambios han sido salvados.

- Si las configuraciones de la PC servidora han sufrido cambios se le muestra al usuario el mensaje: Por alguna razón uno o más archivos de configuración han sido modificados en la PC servidora. ¿Qué desea hacer? Las tres opciones posibles son: aplicar los cambios locales en el servidor; descartar los cambios locales y restablecer la configuración del servidor; aplicar la última configuración válida que se realizó desde la herramienta. El usuario debe seleccionar una de las tres opciones anteriores y luego seleccionar la opción Aceptar para guardar los cambios o la opción Cancelar para mantener la configuración anterior.

**Descartar cambios locales:** la funcionalidad comienza cuando el usuario selecciona la opción Descartar. Permite descartar las configuraciones locales, cambiando estas por las configuraciones de la PC servidora. Luego al usuario se le muestra el mensaje: ¿Está seguro que desea descartar los cambios realizados hasta el momento? Esto causará pérdida en los cambios realizados. Posteriormente el usuario debe seleccionar la opción Aceptar para guardar los cambios o la opción Cancelar para mantener la configuración anterior (MOLINA, 2016).

A continuación se describe la arquitectura de la herramienta.

### 2.2.2 Arquitectura

La herramienta presenta una arquitectura N-Capas orientada al dominio, distribuida en cinco componentes o paquetes: Presentación, Aplicación, Dominio, Persistencia e Infraestructura Transversal, como se aprecia en la Figura 7. La interacción entre los mismos se realiza a través de interfaces y utilizando inyección de dependencias (PALMA, 2013). La capa Dominio es el corazón del software, sus componentes implementan la funcionalidad principal del sistema y encapsulan toda la lógica de negocio relevante (DE LA TORRE, et al., 2010).

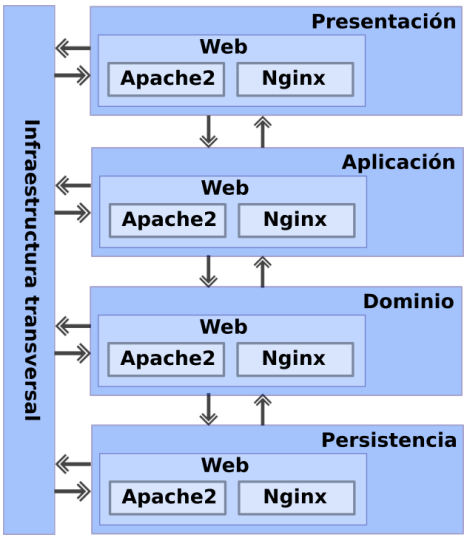


*Figura 7: Arquitectura de HMAST (Fuente: elaboración propia).*

En las capas de Presentación, Aplicación, Dominio y Persistencia se inserta un paquete llamado Web que a su vez contiene dos paquetes, Apache2 y Nginx, con todo lo referente a cada módulo respectivamente.

### 2.2.2 Solución para la selección de Apache 2 y Nginx

Como solución de la presente investigación, en lugar de tener los módulos Apache 2 y Nginx aisladamente, se inserta en HMAST un componente Web que contiene dentro ambos módulos, que inicialmente están desactivados. Cuando se accede por primera vez al componente, se deben completar los campos en la opción *Seleccionar servidor web*, los cuales se muestran a continuación en la *Tabla 13*.

*Tabla 13: Elementos a especificar para la selección del servidor web (Fuente: elaboración propia).*

| **Nombre del campo** | **Valor permitido** |
|---|---|
| Cantidad total de usuarios que acceden al contenido | (número entero positivo) |
| Tipo de contenido | Estático/Dinámico |
| En caso de ser dinámico especificar lenguaje de programación | PHP/Python |
| Medida de la eficiencia que desea priorizar | Todos/Rendimiento/Utilización de los recursos/Capacidad/ |

Posteriormente se presiona el botón *Mostrar resultado*, la herramienta tomando como referencia los 60 escenarios obtenidos en la *Tabla 14*, visualiza el nombre del servidor web más indicado según los elementos especificados anteriormente. Se muestra la opción *Instalar servidor web*, que permite instalar el servidor seleccionado o ambos servidores en caso de ser Proxy la opción resultante. En este último caso, el sistema instala primero Apache 2 y se habilita solo el puerto 8080 escuchando por 127.0.0.1 para posteriormente instalar Nginx y que no existan conflictos. El usuario puede desinstalar el servidor web en el momento deseado.

*Tabla 14: Tabla para la selección del servidor web (Fuente: elaboración propia).*

| **Cantidad de peticiones concurrentes (p)** | **Indicadores** | **Tipo de contenido** | | |
|---|---|---|---|---|
| | | **Estático** | **Dinámico** | |
| | | | **PHP** | **Python** |
| **p≤500** | Rendimiento | Nginx | Nginx | Nginx |
| | Recursos | Nginx | Nginx | Nginx |
| | Capacidad | Apache | Nginx | Proxy |
| | **Todos** | **Nginx** | **Nginx** | **Nginx** |
| **500<p≤1000** | Rendimiento | Nginx | Nginx | Proxy |
| | Recursos | Nginx | Nginx | Nginx |

| | | | | |
|---|---|---|---|---|
| | Capacidad | Apache | Nginx | Nginx/Proxy |
| | **Todos** | **Nginx** | **Nginx** | **Nginx** |
| **1000<p≤10000** | Rendimiento | Apache | Proxy | Proxy |
| | Recursos | Nginx | Nginx | Nginx |
| | Capacidad | Nginx | Proxy | Nginx/Proxy |
| | **Todos** | **Nginx** | **Proxy** | **Nginx** |
| **p>10000** | Rendimiento | Nginx | Proxy | Proxy |
| | Recursos | Nginx | Nginx | Nginx |
| | Capacidad | Apache/Nginx | Nginx | Nginx/Proxy |
| | **Todos** | **Nginx** | **Nginx** | **Nginx** |
| **Todas** | Rendimiento | Nginx | Proxy | Proxy |
| | Recursos | Nginx | Nginx | Nginx |
| | Capacidad | Apache/Nginx | Nginx | Nginx/Proxy |
| | **Todos** | **Nginx** | **Nginx** | **Nginx** |

Como se aprecia en la anterior tabla, teniendo en cuenta todos los indicadores, el servidor web seleccionado en todos los escenarios es Nginx excepto para contenido dinámico con PHP y concurrencia de peticiones entre 1 000 y 10 000, que el seleccionado es Proxy. En cuanto al consumo de recursos el más eficiente en todos los casos es Nginx. Teniendo en cuenta el rendimiento y la capacidad, los tres servidores son elegidos respectivamente en algunos escenarios. A continuación se describen los módulos Apache 2 y Nginx.

### 2.2.3 Módulo Apache 2

En el módulo Apache 2 se pueden administrar las tres secciones de configuración: entorno global, servidor principal y hosts virtuales. A continuación se describen cada una de las funcionalidades.

**Especificar estado del servidor:** las acciones de iniciar, reiniciar y detener el servidor web se realizan en el momento que el usuario lo desee. En el caso de recargar se realiza de forma automática después de aplicar los cambios a la PC servidora.

**Instalar MPM a usar:** uno de los aspectos relevantes que le permite a Apache 2 adaptarse a los diferentes entornos existentes es la instalación del MPM a usar. La interfaz muestra los tres MPM que trae Apache 2 por defecto (*prefork*, *worker* y *event*), lo que se debe seleccionar cuál es el que estará cargado, que debe ser solo uno.

**Especificar parámetros en los MPM *prefork*, *worker* y *event*:** se pueden gestionar los parámetros de cada uno de los MPM. En el caso de *prefok* se especifica el número de procesos hijos que se crean al iniciar Apache; los números mínimo y máximo de procesos hijos inactivos; el número máximo de

procesos hijos que pueden crearse; el número máximo de peticiones que un proceso hijo atenderá durante su existencia donde el valor 0 es para ilimitadas peticiones.

**Especificar parámetros en el servidor principal:** se especifican parámetros como la dirección de correo del administrador; la información sobre versión del servidor, que es la información que se devuelve dentro de la cabecera HTTP que envía el servidor, donde los posibles valores de menor a mayor información son Prod, Min, Os y Full; y la ruta del fichero de registro de errores.

**Especificar parámetros en el manejo de conexiones:** se pueden modificar parámetros como el tiempo que Apache esperará antes de cerrar la conexión; el estado de la utilización de conexiones HTTP persistentes; el tiempo que el servidor esperará peticiones subsiguientes en conexiones persistentes y el número de solicitudes permitidas por conexión donde la cantidad es ilimitada para valor 0.

**Especificar módulos a usar en Apache 2:** se muestra una lista con todos los módulos existentes en Apache y de ellos están marcados los habilitados. Para habilitar o deshabilitar uno o más módulos, deben seleccionarse y presionar el botón *Habilitar* o *Deshabilitar* respectivamente.

**Modificar puertos para las conexiones:** la modificación de los puertos utilizados para las conexiones en Apache 2 se realizará de forma implícita cuando se guarden todos los cambios realizados, una vez que haya finalizado la gestión de todos los hosts virtuales. Se adicionarán todos los puertos que hayan sido asignados a algún host virtual, en caso que no se encuentren en la lista de puertos de Apache 2.

**Especificar permisos de acceso a directorios y ficheros:** se listan los directorios y ficheros a los que se les han establecido permisos de control de acceso. En cada directorio o fichero se especifica si se permite o deniega el acceso desde diferentes orígenes, donde cada uno puede ser un IP completo, un IP parcial, un nombre de dominio completo, un nombre de dominio parcial o una combinación IP/máscara.

**Gestionar hosts virtuales:** la funcionalidad de mayor impacto es gestionar hosts virtuales, la cual facilita un mejor entendimiento por parte de los administradores para la gestión de estos debido a que se muestra de forma detallada la estructura de almacenamiento lógico de los mismos. Apache soporta hosts virtuales basados en nombre y/o hosts virtuales basados en IP. Una PC servidora puede tener asignadas una o varias direcciones IP y cada una de ellas puede escuchar por una lista de puertos específicos y además por todos los puertos, que en la configuración se representa con el símbolo de asterisco (*). A cada combinación *direcciónIP:puerto* se puede asignar un host virtual basado en IP o una lista de hosts virtuales basados en nombre. Para gestionar los hosts virtuales primeramente se listan todos los asignados a la PC servidora, con un filtro para la búsqueda en dependencia de la

dirección a la que están asignados (todos los IP, un IP específico o un nombre de dominio). Los hosts se agrupan por los puertos de escucha. También se muestra si están habilitados o deshabilitados. Con los botones que aparecen en la parte superior derecha del listado se podrán adicionar nuevos hosts, editar alguno de los existentes, eliminar, activar o desactivar uno o varios.

**Adicionar host virtual:** para la adición de un host virtual existen tres grupos de parámetros: *Generales*, *Permisos de acceso* y *Otros parámetros*. En *Generales* como campos obligatorios están el nombre del fichero que contendrá la configuración del host virtual que es el identificador del host pues no puede repetirse y el directorio raíz que es donde se almacena el contenido a publicar por ese host virtual. Además se requiere especificar una lista con las combinaciones *dirección:puerto*, donde dirección puede ser un IP específico, un nombre de dominio, todos los IP (se representa con el símbolo de asterisco) o por defecto (se representa como _default_); el puerto puede ser un número entre 1 y 65535 o todos (se representa con el símbolo de asterisco). En caso que Nginx funcione como proxy inverso de Apache 2, la combinación especificada es 127.0.0.1:8080. Se detallan también parámetros asociados con el soporte SSL y los errores. En *Permisos de acceso* se listan los directorios y ficheros a los que se les han establecido permisos de control de acceso. En *Otros parámetros* se gestionan alias del nombre, archivos índices y alias de directorios o ficheros.

### 2.2.4 Módulo Nginx

El módulo cuenta con las funcionalidades necesarias para configurar las conexiones en el servidor, controlar el tráfico HTTP, gestionar los hosts virtuales, configurar los permisos de acceso y alias para cada host virtual. A continuación se describen cada una de las funcionalidades (MOLINA, 2017):

**Especificar estado del servidor:** el sistema permite al usuario iniciar, detener, reiniciar o recargar el servidor web Nginx. Para ejecutar iniciar, el servicio debe estar detenido, por otra parte para detener o reiniciar, el servicio debe estar ejecutándose. La acción recargar se ejecuta automáticamente después de aplicar los cambios a la PC servidora.

**Gestionar hosts virtuales:** el sistema permite al usuario mostrar el listado de todos los hosts virtuales existentes en el servidor web Nginx, en caso de estar vacío, se muestra un mensaje notificándolo. Para adicionar un host virtual los parámetros obligatorios son el nombre del fichero que contendrá la configuración del host virtual; la dirección IP que puede ser un IP específico de los asignados a la PC servidora o un asterisco (*) en caso de que se defina acceder a un host virtual desde cualquier dirección IP; el puerto de escucha asignado al host virtual que debe ser un número entero positivo entre 1 y 65535; el directorio raíz el cual almacena el contenido a publicar y el nombre del servidor que se refiere al nombre de dominio asignado al host virtual. Además se pueden especificar los archivos índices del directorio raíz del host virtual. Otro elemento es si Nginx funcionará como proxy inverso de Apache 2,

con argumento on/off respectivamente, por defecto es (off), en caso que el usuario especifique (on), el sistema realiza la configuración en el host virtual para enviar el contenido dinámico a Apache 2, tomando el lenguaje de programación utilizado de los elementos especificados en la selección.

Una vez introducidos dichos parámetros se crea en la PC donde está instalada HMAST, el fichero de configuración del host. Para realizar la edición debe existir al menos un host virtual en el listado, posteriormente se selecciona uno y se presiona el botón *Editar* para que muestre la interfaz de edición. Para eliminar uno o varios hosts virtuales el usuario selecciona el/los que desea eliminar, una vez seleccionados se presiona el botón *Eliminar*, se muestra un mensaje de confirmación. El sistema permite al usuario habilitar o deshabilitar un host virtual, el usuario accede a la lista de los hosts virtuales y selecciona el/los que desea habilitar/deshabilitar. Posteriormente presiona el botón correspondiente para realizar la acción deseada, donde se muestra un mensaje de confirmación.

**Especificar parámetros para las conexiones:** el sistema permite al usuario especificar los parámetros referentes a las conexiones: el número máximo de conexiones simultáneas por procesos; si el servidor acepta o no varias peticiones por conexión, con argumento on/off que significa que el servidor acepta conexiones persistentes (on) o no acepta conexiones persistentes (off); la cantidad de segundos entre peticiones y el número máximo de procesos, si el valor de dicha directiva es auto, significa que el servidor intenta detectar de forma automática el número de núcleos disponibles de la CPU.

Otras funcionalidades son las referentes a la gestión de los directorios donde se almacena la información que va a publicar un host virtual, el sistema permite listar, adicionar, editar, mostrar y eliminar un directorio. El sistema permite además mostrar y editar las configuraciones del tráfico HTTP.

### 2.3 Conclusiones parciales

Al finalizar el presente capítulo se concluye lo siguiente:

- El empleo de un estudio de caso aplicando la Norma Cubana ISO/IEC 25023:2017 permitió definir la teoría relacionada con la eficiencia de los servidores web Apache 2 y Nginx,
- El cálculo de la eficiencia a partir del tipo de contenido y la cantidad de peticiones concurrentes permitió la definición de 60 escenarios para la selección del servidor web, resultando ser Nginx el más eficiente en 47 de ellos, Apache en 5 y Nginx funcionando como Proxy inverso de Apache en 14.
- La integración de los 60 escenarios definidos a la Herramienta para la Migración y Administración de Servicios Telemáticos, permitió definir y desarrollar un componente Web que selecciona el servidor web más eficiente en cuanto a las características de la institución.

# CAPÍTULO 3. EVALUACIÓN DE LA SOLUCIÓN

La evaluación de la solución tiene la finalidad de verificar el cumplimiento del objetivo propuesto, que es aumentar la eficiencia en la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto. En el presente capítulo a partir de la definición de eficiencia, se evalúa la solución con la realización de un estudio de caso midiendo cada uno de los indicadores de la eficiencia de desempeño. Además se aplica la técnica de Iadov para determinar el nivel de satisfacción de los especialistas en servicios telemáticos y se evalúa la aplicabilidad de la solución con el criterio de expertos a través del método Delphi.

## 3.1 Eficiencia

La eficiencia, según el Diccionario de la Lengua Española (RAE, 2018), es la capacidad de disponer de alguien o de algo para conseguir un efecto determinado. Constituye la capacidad de alcanzar un objetivo determinado mediante una utilización racional de los recursos disponibles, sean materiales o tiempo, o sea, con el mínimo de recursos posibles y/o en el menor tiempo posible (PARTIDO COMUNISTA DE CUBA, 2017).

En cuanto a la calidad interna y externa del software, la eficiencia es una de las seis categorías que constituyen el modelo planteado por la Norma Cubana ISO/IEC 9126 del año 2005. Esta norma plantea que la calidad del producto del software se debe evaluar usando un modelo de calidad definido. La norma define la eficiencia como la capacidad del producto de software para proporcionar una ejecución o desempeño apropiado, en relación con la cantidad de recursos utilizados usados, bajo condiciones establecidas (OFICINA NACIONAL DE NORMALIZACIÓN, 2005).

La norma ISO/IEC 9126 fue cancelada y reemplazada por la ISO/IEC 25023:2017, que establece las medidas de calidad del producto de software y del sistema, donde la eficiencia de desempeño es una de ellas. Las medidas de eficiencia de desempeño se utilizan para evaluar el desempeño en relación con la cantidad de recursos utilizados en condiciones establecidas. Los recursos pueden incluir otros productos de software, la configuración de software y hardware del sistema y materiales (por ejemplo, papel de impresión, medios de almacenamiento). Se ve fuertemente afectada y fluctúa dependiendo de las condiciones de uso, como la carga de datos de procesamiento, la frecuencia de uso, el número de sitios de conexión, entre otros. La eficiencia de desempeño se mide en cuanto a los siguientes parámetros (OFICINA NACIONAL DE NORMALIZACIÓN, 2017):

- Rendimiento: Las medidas de rendimiento se utilizan para evaluar el grado en que la respuesta y los tiempos de procesamiento y las tasas de rendimiento de un producto o sistema al realizar sus funciones cumplen los requisitos.

- Utilización de los recursos: Las medidas de utilización de los recursos se emplean para evaluar el grado en que las cantidades y los tipos de recursos utilizados por un producto o sistema en el desempeño de sus funciones cumplen los requisitos.
- Capacidad: Las medidas de capacidad se utilizan para evaluar el grado en que los límites máximos de un producto o parámetro del sistema cumplen los requisitos.

En el Anexo 3 aparece la operacionalización de la variable eficiencia de desempeño, donde se muestran los indicadores para la medición de la misma, agrupados según los parámetros que plantea la Norma Cubana ISO/IEC 25023:2017. A continuación se realiza un estudio de caso para evaluar la eficiencia obtenida con la aplicación de la solución en ECOAIND3.

### 3.2 Estudio de caso en ECOAIND3

La Empresa Constructora de Obras de Arquitectura e Industriales No. 3, Contingente “Ñico López” lleva 29 años dedicada a la construcción y reparación de obras, en programas de gran importancia para la economía del país, como las obras del turismo, las de Biocubafarma, las de viviendas y sobre todo las relacionadas con el polo científico y de otros programas priorizados para la economía. La empresa presenta una estructura compuesta por una Oficina Central que organiza, controla, regula y dirige todo el proceso constructivo, y 5 unidades empresariales de base. En la Oficina Central se encuentran los servidores que ofrecen los servicios telemáticos a los 200 usuarios que posee la empresa. Existe una PC servidora (ver características en la Tabla 15) que tiene instalado el servidor web que publica el repositorio de Nova y la actualización del antivirus (Nod32, Kaspersky, Clamav, Segurmática).

*Tabla 15: Características del escenario de prueba.*

| RAM | CPU | HDD | Tipo de contenido publicado |
|---|---|---|---|
| 4 GB | Intel Core i5 3.20 GHz y 4 procesadores | 32 GB | Estático |

En ECOAIND 3 se ejecutó el proyecto de migración a aplicaciones de código abierto desde julio de 2014 a marzo de 2015, efectuado por el Centro de Software Libre de la UCI. En abril de 2018 comenzó el proyecto de actualización a la migración de los servidores, que concluyó en diciembre del propio año. En el primer proyecto, la selección del servidor web a instalar se realizó sin una guía que orientara cuál era el indicado según las características de la empresa y en el segundo proyecto se utilizó la solución resultante de la presente investigación. A partir del análisis de ambos proyectos es que se realiza el presente estudio de caso, cuyas características de describen a continuación.

**Objetivo del estudio:** Comparar la eficiencia del servidor web seleccionado en cada proyecto de migración a código abierto.

**Unidad de análisis:** eficiencia del servidor web.

**Descripción:** para la medición de la eficiencia se empleó la Norma Cubana ISO/IEC 25023:2017, las fórmulas utilizadas aparecen en el Anexo 3. Se realizaron cuatro observaciones, utilizando la herramienta *ab* mediante la cual se crearon un total de 200 peticiones con concurrencias 10, 100, 150 y 200 respectivamente, desde una PC cliente al servidor web instalado en la PC servidora. Al mismo tiempo se realizó el monitoreo de los recursos de la PC servidora con la herramienta *dstat*.

### 3.2.1 Análisis del proyecto realizado en 2015

Como resultado del proceso de migración de los servicios telemáticos, el servidor web seleccionado fue Apache 2 porque era del que más conocimientos tenían los especialistas en servicios telemáticos y los administradores de redes de la empresa. Todas las PC de la empresa tienen una tarea programada con el servicio cron, que actualiza el antivirus diariamente descargando la actualización que publica el servidor web. La tarea programada para la actualización del antivirus, tiene horarios diferentes para que todos los usuarios no accedan simultáneamente en el mismo instante de tiempo. En la configuración del servidor web solo permite hasta 150 peticiones concurrentes, lo que prohíbe que todos los usuarios accedan a la actualización del antivirus simultáneamente. Se detectó que el servidor web consume muchos recursos de hardware y por tanto su funcionamiento es lento, lo que trae consigo lentitud en responder a las peticiones de los usuarios.

### 3.2.2 Análisis del proyecto realizado en 2018

Para la actualización de los servidores en el año 2018, los especialistas en servicios telemáticos instalaron la Herramienta para la Migración y Administración de Servicios Telemáticos en una PC cliente, desde la cual accedieron a la PC servidora. Primeramente detuvieron el servicio *apache2* para que no hubieran conflictos y poder efectuar la selección del servidor web según las características de la empresa. Después de especificar 200 como el total de peticiones concurrentes, tipo de contenido estático y Todos como la medida de eficiencia a priorizar, el servidor web seleccionado fue Nginx, el cual se instaló y configuró para el acceso simultáneo de todos los usuarios de la empresa.

### 3.2.3 Análisis de la eficiencia en ambos proyectos

Como resultado en ambos proyectos, se calculó la eficiencia del servidor web correspondiente en cuanto a todos los indicadores, las fórmulas se encuentran en el Anexo 3.

**Tiempo medio de conclusión de un trabajo:** se calcula como la sumatoria del tiempo que demora el servidor web en dar respuesta a una petición en cada observación (expresado en milisegundos[6]),

[6] Un milisegundo es equivalente a 0,001 segundo y un segundo equivale a 1000 milisegundos.

dividido entre la cantidad de observaciones. A continuación se muestra el cálculo de la eficiencia para ambos servidores web, un valor menor es mejor.

*Cálculo de la eficiencia para Apache 2*

$$X = \frac{547{,}9 + 610{,}3 + 537{,}7 + 553{,}1}{4} = \frac{2249}{4} = 562{,}250$$

*Cálculo de la eficiencia para Nginx*

$$X = \frac{180{,}2 + 186{,}6 + 195{,}1 + 199{,}1}{4} = \frac{761}{4} = 190{,}250$$

**Adecuación del tiempo de conclusión de un trabajo:** se calcula como el cociente entre el tiempo medio de conclusión de un trabajo medido por el indicador anterior y el tiempo de respuesta especificado, que es 200 ms. A continuación se muestra el cálculo de la eficiencia para ambos servidores web, un valor menor es mejor.

*Cálculo de la eficiencia para Apache 2*

$$X = \frac{562{,}250}{200} = 2{,}811$$

*Cálculo de la eficiencia para Nginx*

$$X = \frac{190{,}250}{200} = 0{,}951$$

**Rendimiento medio:** se calcula como la sumatoria del cociente entre el número de peticiones y el tiempo de la observación (expresado en segundos), dividido por el número de observaciones. A continuación se muestra el cálculo de la eficiencia para ambos servidores web, un valor mayor es mejor.

*Cálculo de la eficiencia para Apache 2*

$$X = \frac{\frac{200}{109{,}575} + \frac{200}{122{,}053} + \frac{200}{107{,}548} + \frac{200}{110{,}628}}{4}$$

$$X = \frac{1{,}825 + 1{,}639 + 1{,}860 + 1{,}808}{4} = \frac{7{,}131}{4} = 1{,}783$$

*Cálculo de la eficiencia para Nginx*

$$X = \frac{\frac{200}{36{,}040} + \frac{200}{37{,}320} + \frac{200}{39{,}020} + \frac{200}{39{,}820}}{4}$$

$$X = \frac{5{,}549 + 5{,}359 + 5{,}126 + 5{,}023}{4} = \frac{21{,}057}{4} = 5{,}264$$

**La media de utilización del procesador:** se calcula como el cociente entre el tiempo del procesador realmente utilizado (ver Figura 8) y el tiempo de operación (ambos expresados en milisegundos), dividido por el número de observaciones.

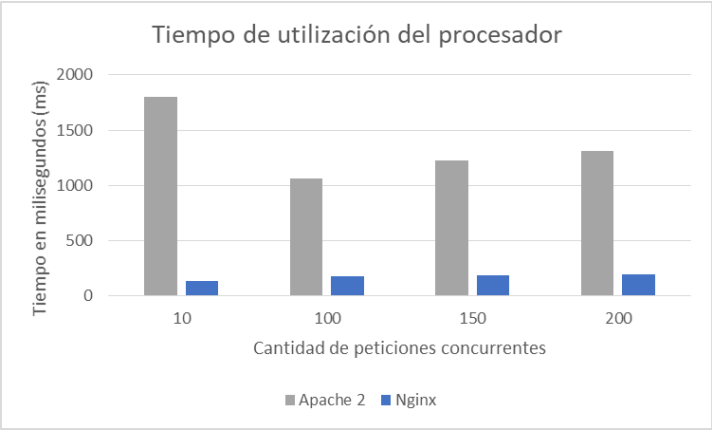


*Figura 8: Tiempo de utilización del procesador en ECOAIND3 (Fuente: elaboración propia).*

Como se aprecia en la anterior figura, Nginx utiliza mucho menos tiempo del procesador que Apache 2. A continuación se muestra el cálculo de la eficiencia para ambos servidores web, un valor menor es mejor.

*Cálculo de la eficiencia para Apache 2*

$$X = \frac{\frac{1800,8}{109575} + \frac{1065,9}{122053} + \frac{1221,3}{107548} + \frac{1314,0}{110628}}{4}$$

$$X = \frac{0,016 + 0,009 + 0,011 + 0,012}{4} = \frac{0,048}{4} = 0,012$$

*Cálculo de la eficiencia para Nginx*

$$X = \frac{\frac{130,6}{36040} + \frac{180,1}{37320} + \frac{182,2}{39020} + \frac{193,4}{39820}}{4}$$

$$X = \frac{0,004 + 0,005 + 0,005 + 0,005}{4} = \frac{0,019}{4} = 0,005$$

**La media de utilización de la memoria:** se calcula como la sumatoria del cociente entre el consumo de RAM (ver Figura 9) y la memoria disponible que es la capacidad total de RAM de la PC servidora (4096 MB), dividido por el número de observaciones.

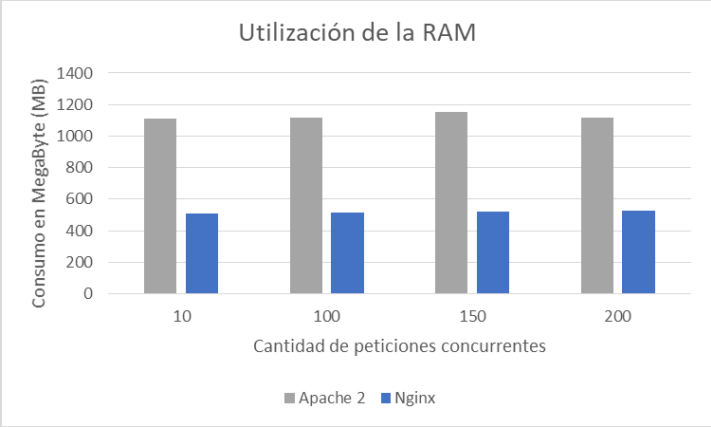


*Figura 9: Utilización de la RAM en ECOAIND3 (Fuente: elaboración propia).*

Como se aprecia en la anterior figura, el consumo de RAM por Nginx es menor que el de Apache 2. A continuación se muestra el cálculo de la eficiencia para ambos servidores web, un valor menor es mejor.

*Cálculo de la eficiencia para Apache 2*

$$X = \frac{\frac{1110,6}{4096} + \frac{1115,3}{4096} + \frac{1154,3}{4096} + \frac{1167,8}{4096}}{4}$$

$$X = \frac{0,271 + 0,272 + 0,282 + 0,273}{4} = \frac{1,099}{4} = 0,275$$

*Cálculo de la eficiencia para Nginx*

$$X = \frac{\frac{507,3}{4096} + \frac{512,6}{4096} + \frac{523,7}{4096} + \frac{528,4}{4096}}{4}$$

$$X = \frac{0,124 + 0,125 + 0,128 + 0,129}{4} = \frac{0,506}{4} = 0,126$$

**La media del uso de los dispositivos de E/S:** se calcula como la sumatoria del cociente entre el tiempo ocupado de los dispositivos y la duración de las operaciones de E/S, dividido entre la cantidad de observaciones. Aplicando la regla de tres se sustituyó en la fórmula el cociente entre el porciento y 100, en lugar del cociente entre el tiempo ocupado de los dispositivos y la duración de las operaciones

de E/S. A continuación se muestra el cálculo de la eficiencia para ambos servidores web, un valor menor es mejor.

*Cálculo de la eficiencia para Apache 2*

$$X = \frac{0{,}090 + 0{,}096 + 0{,}093 + 0{,}082}{4} = \frac{0{,}361}{4} = 0{,}090$$

*Cálculo de la eficiencia para Nginx*

$$X = \frac{0{,}091 + 0{,}087 + 0{,}094 + 0{,}089}{4} = \frac{0{,}361}{4} = 0{,}090$$

**Utilización del ancho de banda:** se calcula como el cociente entre el ancho de banda de transmisión real medido a lo largo del tiempo y la capacidad de ancho de banda disponible (10 Mbps). A continuación se muestra el cálculo de la eficiencia para ambos servidores web, un valor menor es mejor.

*Cálculo de la eficiencia para Apache 2*

$$X = \frac{6{,}470}{10} = 0{,}647$$

*Cálculo de la eficiencia para Nginx*

$$X = \frac{5{,}430}{10} = 0{,}543$$

**Capacidad de procesamiento de transacciones:** se calcula como el cociente entre la cantidad de transacciones y el tiempo de la observación, que se obtuvo con la cantidad de transacciones de disco por segundo (tps). A continuación se muestra el cálculo de la eficiencia para ambos servidores web, un valor menor es mejor.

*Cálculo de la eficiencia para Apache 2*

$$X = 2{,}864$$

*Cálculo de la eficiencia para Nginx*

$$X = 2{,}921$$

**Capacidad de acceso de usuario:** se calcula como la sumatoria de la cantidad de usuarios que acceden en cada observación, dividida entre el número de observaciones. A continuación se muestra el cálculo de la eficiencia para ambos servidores web, un valor mayor es mejor.

*Cálculo de la eficiencia para Apache 2*

$$X = \frac{200 + 200 + 200 + 200}{4} = \frac{800}{4} = 200$$

*Cálculo de la eficiencia para Nginx*

$$X = \frac{200 + 200 + 200 + 200}{4} = \frac{800}{4} = 200$$

Finalmente se realizó la comparación de la eficiencia de ambos servidores web en cuanto a todos los indicadores, como se aprecia en la Tabla 16.

*Tabla 16: Comparación de la eficiencia del servidor web en ambos proyectos (Fuente: elaboración propia).*

| Indicadores | Eficiencia de Apache 2 | Eficiencia de Nginx |
|---|---|---|
| Tiempo medio de conclusión de un trabajo (ms) | 562,250 | **190,250** |
| Adecuación del tiempo de conclusión de un trabajo (ms) | 2,811 | **0,951** |
| Rendimiento medio | 1,783 | **5,264** |
| La media de utilización del procesador | 0,012 | **0,005** |
| La media de utilización de la memoria | 0,275 | **0,126** |
| La media del uso de los dispositivos de E/S | 0,090 | 0,090 |
| Utilización del ancho de banda | 0,647 | **0,543** |

| | | |
|---|---|---|
| Capacidad de procesamiento de transacciones | 2,864 | **2,921** |
| Capacidad de acceso de usuario | 200 | 200 |

Como se observa en la Tabla 16, se muestra en negrita el valor resultante del servidor más eficiente, donde en siete indicadores Nginx supera a Apache 2 y en dos indicadores ambos servidores tienen la misma eficiencia. Con lo anteriormente planteado se concluye que con la aplicación de la solución en ECOAIND3 se aumentó la eficiencia en la selección del servidor web.

### 3.3 Aplicación del criterio de expertos con el método Delphi

Existen varios métodos para la obtención de juicios de expertos, que pueden clasificarse según si la evaluación se realiza de manera individual o grupal. En el primer grupo se encuentra el método Delphi (ESCOBAR y CUERVO, 2008), que es un método de consenso que tiene como objetivo encontrar un acuerdo general entre un panel de expertos sobre un tema de investigación específico (GALANIS, 2018). Cada juez realiza la evaluación individualmente, pero luego de analizar las respuestas se le envía a cada juez la mediana obtenida y se le pide que reconsidere su juicio hasta que se logre un consenso. Esta técnica ofrece un alto nivel de interacción entre los expertos (ESCOBAR y CUERVO, 2008).

#### 3.3.1 Selección de los expertos

Los expertos son considerados como personas que tienen una estrecha relación sobre la cuestión, sector, tecnología u objeto de la investigación (GARCÍA y LENA, 2018), en este caso con los servidores web. Para la selección de los expertos se aplicó una encuesta que se puede apreciar en el Anexo 6, a 10 especialistas que han desempeñado el rol de especialistas en servicios telemáticos en el Centro de Software Libre de la UCI. Se tuvieron en consideración los siguientes elementos: título universitario, categoría científica y docente, años de experiencia en el desarrollo de procesos de migración, el conocimiento acerca de los servidores web y las fuentes de argumentación.

Posteriormente se procedió a calcular el Coeficiente de Competencia Experta $K$, calculado a partir de la opinión mostrada por el propio experto sobre su nivel de conocimiento acerca del problema analizado, así como de las fuentes que permiten argumentar su respuesta. En dicho coeficiente se promedian dos factores, el Coeficiente de Conocimiento $Kc$ y el Coeficiente de Argumentación $Ka$, a partir de la fórmula $K = (Kc + Ka) * 0{,}5$ (GARCÍA y LENA, 2018).

El Coeficiente de Conocimiento $Kc$ se determinó (ver Tabla 17) a partir de la información que el propio experto presentó en una escala del 0 al 10, donde 10 implica pleno conocimiento sobre los servidores web y el valor 0 el nulo conocimiento del tema, multiplicado por 0,1. En la Tabla 18 se muestra un resumen de la cantidad de expertos por cada Coeficiente de Conocimiento.

*Tabla 17: Cálculo del Coeficiente de Conocimiento de los expertos (Fuente: elaboración propia).*

| Número de experto | Escala | | | | | | | | | | Kc |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
| **1** | | | | | | | x | | | | **0,7** |
| **2** | | | | | | | | x | | | **0,8** |
| **3** | | | | | | | | | x | | **0,9** |
| **4** | | | | | | | | x | | | **0,8** |
| **5** | | | | | | | x | | | | **0,7** |
| **6** | | | | | | | | x | | | **0,8** |
| **7** | | | | | | | | | | x | **1,0** |
| **8** | | | | | | | | | x | | **0,9** |
| **9** | | | | | | | x | | | | **0,7** |
| **10** | | | | | | x | | | | | **0,6** |

*Tabla 18: Resumen de la ubicación de los expertos según Kc (Fuente: elaboración propia).*

| $Kc$ | 1,0 | 0,9 | 0,8 | 0,7 | 0,6 |
|---|---|---|---|---|---|
| **Cantidad de expertos** | 1 | 2 | 3 | 3 | 1 |

Para calcular el Coeficiente de Argumentación, primeramente se sometieron a evaluación las fuentes que le han posibilitado al experto enriquecer su conocimiento sobre el tema, asignando en cada caso una de las categorías: Alta (M), Media (M) o Baja (B). Esas evaluaciones cualitativas se ponderaron con valores numéricos (ver Tabla 19).

*Tabla 19: Grado de influencia de las fuentes de argumentación (Fuente: elaboración propia).*

| No. | Fuentes de argumentación | Alto (A) | Medio (M) | Bajo (B) |
|---|---|---|---|---|
| 1 | Estudios teóricos realizados por usted | 0,30 | 0,20 | 0,10 |
| 2 | Experiencia adquirida durante su vida profesional | 0,50 | 0,40 | 0,30 |
| 3 | Conocimiento de investigaciones y/o publicaciones nacionales e internacionales. | 0,05 | 0,04 | 0,03 |
| 4 | Conocimiento propio sobre el estado del tema de investigación. | 0,05 | 0,04 | 0,03 |
| 5 | Actualización en cursos de posgrado, diplomados, maestrías, doctorado. | 0,05 | 0,04 | 0,03 |
| 6 | Intuición | 0,05 | 0,04 | 0,03 |

Finalmente se calculó $Ka$ sumando por cada experto, los valores numéricos de cada fuente de argumentación (ver Tabla 20).

*Tabla 20: Matriz de Coeficientes de Argumentación por expertos (Fuente: elaboración propia).*

| Número de exp | Fuente de argumentación | | | | | | Ka |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | |
| **1** | 0,3 | 0,4 | 0,05 | 0,05 | 0,05 | 0,05 | **0,9** |
| **2** | 0,3 | 0,5 | 0,04 | 0,04 | 0,05 | 0,05 | **0,98** |
| **3** | 0,3 | 0,5 | 0,05 | 0,05 | 0,05 | 0,04 | **0,99** |
| **4** | 0,2 | 0,5 | 0,03 | 0,05 | 0,04 | 0,03 | **0,85** |
| **5** | 0,2 | 0,5 | 0,05 | 0,04 | 0,05 | 0,05 | **0,89** |
| **6** | 0,3 | 0,4 | 0,05 | 0,05 | 0,03 | 0,05 | **0,88** |
| **7** | 0,2 | 0,5 | 0,05 | 0,05 | 0,05 | 0,05 | **0,9** |
| **8** | 0,3 | 0,4 | 0,04 | 0,05 | 0,05 | 0,05 | **0,89** |
| **9** | 0,3 | 0,5 | 0,05 | 0,04 | 0,04 | 0,04 | **0,97** |
| **10** | 0,2 | 0,5 | 0,04 | 0,04 | 0,05 | 0,04 | **0,87** |

A partir de $Kc$ y $Ka$, utilizando la fórmula anteriormente mencionada, se obtuvo el Coeficiente de Competencia Experta $K$. En cada caso se analizó $K$ y ubicándolo en el intervalo correspondiente según la Tabla 21, se determinó el nivel de competencia de cada experto, donde el 90% posee un nivel de competencia Alto y un 10% un nivel Medio, como se aprecia en la Tabla 22.

*Tabla 21: Intervalos para definir la competencia de un experto (GARCÍA y LENA, 2018).*

| Nivel de competencia | | |
|---|---|---|
| **Alto (A)** | **Medio (M)** | **Bajo (B)** |
| 1 ≥ K ≥0,8 | 0,8 > K ≥ 0,5 | K < 0,5 |

*Tabla 22: Niveles de competencia de los expertos (Fuente: elaboración propia).*

| Número de experto | Kc | Ka | K | Nivel de competencia |
|---|---|---|---|---|
| **1** | 0,7 | 0,9 | 0,8 | **Alto** |
| **2** | 0,8 | 0,98 | 0,89 | **Alto** |
| **3** | 0,9 | 0,99 | 0,95 | **Alto** |
| **4** | 0,8 | 0,85 | 0,83 | **Alto** |
| **5** | 0,7 | 0,89 | 0,8 | **Alto** |
| **6** | 0,8 | 0,88 | 0,84 | **Alto** |
| **7** | 1,0 | 0,9 | 0,95 | **Alto** |
| **8** | 0,9 | 0,89 | 0,9 | **Alto** |
| **9** | 0,7 | 0,97 | 0,84 | **Alto** |
| **10** | 0,6 | 0,87 | 0,74 | **Medio** |

### 3.3.2 Valoración de la solución por los expertos seleccionados

El método Delphi tiene el propósito de lograr el mayor consenso posible entre los expertos implicados, de forma empírica se considerará que se ha alcanzado el mismo determinándolo a través de la medida de la varianza en las respuestas de los panelistas a través de las diferentes rondas (GARCÍA y LENA, 2018). Se realizó el envío a los expertos de un documento que describe la solución para la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto, y una encuesta que aparece en el Anexo 7, donde se solicitó que evaluaran cinco aspectos relacionados con la misma. La evaluación de los aspectos se basó en cinco categorías: Muy Adecuado (MA), Bastante Adecuado (BA), Adecuado (A), Poco Adecuado (PA) y Nada Adecuado (NA), como se aprecia en la Tabla 23.

*Tabla 23: Clasificación asignada por los expertos a los aspectos propuestos (Fuente: elaboración propia).*

| Número de experto | Aspectos | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 1 | MA | MA | BA | MA | MA |
| 2 | MA | BA | MA | MA | BA |
| 3 | MA | MA | MA | BA | MA |
| 4 | BA | BA | BA | A | MA |
| 5 | MA | BA | MA | MA | A |
| 6 | MA | BA | MA | BA | MA |
| 7 | MA | BA | BA | BA | MA |
| 8 | BA | MA | BA | MA | BA |
| 9 | BA | BA | MA | A | MA |
| 10 | MA | MA | MA | MA | BA |

Una de las principales dificultades en la evaluación y validación de cuestionarios mediante el método Delphi proviene de la subjetividad de los criterios esgrimidos por el panel de expertos. Para compensar y equilibrar la subjetividad de las opiniones expresadas por el panel de expertos mediante escalas nominales, se emplea el Modelo de Torgerson, que permite la conversión de la escala original cualitativa en una escala de intervalo (cuantitativa), que permita la valoración de cada uno de los aspectos de forma individual (GARCÍA y LENA, 2018). A partir de las valoraciones en la Tabla 23, se calcularon las frecuencias absolutas por aspecto (ver Tabla 24) y las frecuencias acumuladas (ver Tabla 25).

*Tabla 24: Frecuencia absoluta por aspecto (Fuente: elaboración propia).*

| Número de aspecto | MA | BA | A | PA | NA | Total |
|---|---|---|---|---|---|---|
| **1** | 7 | 3 | 0 | 0 | 0 | **10** |
| **2** | 4 | 6 | 0 | 0 | 0 | **10** |
| **3** | 6 | 4 | 0 | 0 | 0 | **10** |
| **4** | 5 | 3 | 2 | 0 | 0 | **10** |
| **5** | 6 | 3 | 1 | 0 | 0 | **10** |

*Tabla 25: Frecuencia acumulada por aspecto (Fuente: elaboración propia).*

| Número de aspecto | MA | BA | A | PA | NA |
|---|---|---|---|---|---|
| **1** | 7 | 10 | 10 | 10 | 10 |
| **2** | 4 | 10 | 10 | 10 | 10 |
| **3** | 6 | 10 | 10 | 10 | 10 |
| **4** | 5 | 8 | 10 | 10 | 10 |
| **5** | 6 | 9 | 10 | 10 | 10 |

Posteriormente se calculó la frecuencia acumulada relativa como se muestra en la Tabla 26, obtenida del cociente entre la frecuencia acumulada por aspecto y el número de expertos, expresando esta última con dos cifras decimales (GARCÍA y LENA, 2018).

*Tabla 26: Frecuencia acumulada relativa por aspecto (Fuente: elaboración propia).*

| Número de aspecto | MA | BA | A | PA | NA |
|---|---|---|---|---|---|
| **1** | 0,70 | 1,00 | 1,00 | 1,00 | 1,00 |
| **2** | 0,40 | 1,00 | 1,00 | 1,00 | 1,00 |
| **3** | 0,60 | 1,00 | 1,00 | 1,00 | 1,00 |
| **4** | 0,50 | 0,80 | 1,00 | 1,00 | 1,00 |
| **5** | 0,60 | 0,90 | 1,00 | 1,00 | 1,00 |

Mediante las frecuencias relativas acumuladas se procedió al cálculo de los Puntos de Corte y sus respectivas escalas de indicadores por medio de los valores normales estándar inversos de las propias probabilidades acumuladas de cada indicador en cada experto. Para ello se empleó la aproximación al valor más cercano de la curva Normal Estándar de la probabilidad acumulada. Es necesario indicar que, para los valores de probabilidad acumulada iguales a 1, el valor estándar inverso correspondiente se considera 3,5. De igual forma, para valores de probabilidad acumulada iguales a 0, el valor estándar inverso se asumirá igual a -3,5 (GARCÍA y LENA, 2018). Teniendo en cuenta dicha consideración, el

cálculo se realizó en Excel, introduciendo en la celda A1 el valor a calcular y en la celda A2 insertando la siguiente fórmula:

=SI(A1=1;3,5;SI(A1=0;-3,5;REDONDEAR(DISTR.NORM.ESTAND.INV(A1);2))).

La determinación de los Puntos de Corte se aprecia en la Tabla 27. A los valores calculados se añadió la columna Promedio, obtenida del cálculo del promedio de los valores hallados por fila. De forma similar se estimaron los Puntos de Corte, calculando estos como el promedio de los valores de la función estándar inversa por cada columna. Se determinó también el valor de Límite N, a través del promedio de los Puntos de Corte (cuyo resultado será el mismo que el promedio de los promedios de cada categoría) y que delimitarán los verdaderos rangos de intervalo a los que pertenece cada categoría (GARCÍA y LENA, 2018).

*Tabla 27: Determinación de los Puntos de Corte (Fuente: elaboración propia).*

| Aspecto | 5 (MA) | 4 (BA) | 3 (A) | 2 (PA) | 1 (NA) | Promedio (P) | Diferencia (N-P) | Clasificación |
|---|---|---|---|---|---|---|---|---|
| **1** | 0,52 | 3,50 | 3,50 | 3,50 | 3,50 | **2,90** | **-0,27** | **MA** |
| **2** | -0,25 | 3,50 | 3,50 | 3,50 | 3,50 | **2,75** | **-0,12** | **MA** |
| **3** | 0,25 | 3,50 | 3,50 | 3,50 | 3,50 | **2,85** | **-0,22** | **MA** |
| **4** | 0 | 0,84 | 3,50 | 3,50 | 3,50 | **2,27** | **0,36** | **BA** |
| **5** | 0,25 | 1,28 | 3,50 | 3,50 | 3,50 | **2,41** | **0,22** | **BA** |
| **Puntos de Corte (PC)** | **0,15** | **2,52** | **3,50** | **3,50** | **3,50** | | | |
| | **Valor límite (N = 2,63)** | | | | | | | |

En la Figura 10 se muestra una recta numérica con los Puntos de Corte y los valores de (N-P) en el intervalo correspondiente, clasificando de esta forma cada aspecto sometido a evaluación. Se concluye que de los cinco aspectos, tres se evalúan de Muy Adecuado (60%) y dos de Bastante Adecuado (40%), sin mostrarse ninguno de los casos como Adecuado, Poco adecuado o Nada adecuado.

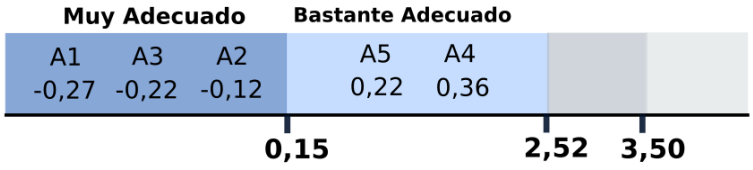


*Figura 10: Determinación de los rangos y Puntos de Corte (Fuente: elaboración propia).*

Para determinar el grado de concordancia de los expertos, los resultados de la Tabla 23 se llevaron a una escala cuantitativa donde 5 = Muy Adecuado (MA), 4 = Bastante Adecuado (BA), 3 = Adecuado (A), 2 = Poco Adecuado (PA) y 1 = Nada Adecuado (NA) (SÁNCHEZ, 2015), como se aprecia en la

Tabla 29. Se procedió a calcular el Coeficiente de Concordancia ($C$) para cada uno de los aspectos, ver fórmulas en la Tabla 28.

*Tabla 28: Variables y fórmulas para calcular el coeficiente de concordancia (Fuente: elaboración propia).*

| **Variable** | Coeficiente de Concordancia | Desviación Estándar | Media del criterio de los expertos por indicador |
|---|---|---|---|
| **Fórmula** | $C = 100 * \left(1 - \frac{Ds}{Xm}\right)$ | $Ds = \sqrt{\frac{1}{n-1}\sum_{i=1}^{10}(Xi - Xm)^2}$ | $Xm = \frac{\sum_{i=1}^{10} CEi}{10}$ |

*Tabla 29: Coeficiente de concordancia por aspecto (Fuente: elaboración propia).*

| **Número de experto** | **Aspectos** | | | | |
|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** |
| **1** | 5 | 5 | 4 | 5 | 5 |
| **2** | 5 | 4 | 5 | 5 | 4 |
| **3** | 5 | 5 | 5 | 4 | 5 |
| **4** | 4 | 4 | 4 | 3 | 5 |
| **5** | 5 | 4 | 5 | 5 | 3 |
| **6** | 5 | 4 | 5 | 4 | 5 |
| **7** | 5 | 4 | 4 | 4 | 5 |
| **8** | 4 | 5 | 4 | 5 | 4 |
| **9** | 4 | 4 | 5 | 3 | 5 |
| **10** | 5 | 5 | 5 | 5 | 4 |
| **Suma** | **47** | **44** | **46** | **43** | **45** |
| **Xm** | **4,7** | **4,4** | **4,6** | **4,3** | **4,4** |
| **Ds** | **0,48** | **0,52** | **0,52** | **0,82** | **0,71** |
| $\boldsymbol{C = 100 * \left(1 - \frac{Ds}{Xm}\right)}$ | **89,79** | **88,18** | **88,70** | **80,93** | **83,86** |

Cada Coeficiente de Concordancia debe tener un valor superior a 75 (SÁNCHEZ, 2015). A partir del cálculo de $\boldsymbol{C}$ se determina que hay un grado de concordancia mayor que 80 en todos los aspectos, por lo que existe un consenso en las evaluaciones dadas por los expertos a los aspectos planteados. A continuación se calcula el Coeficiente de Concordancia Total $Ct$, como se muestra en la Tabla 30, a partir de los votos totales y los votos negativos. Se consideran votos negativos aquellos que se refieren a un aspecto con la categoría de Poco Adecuado o Nada Adecuado (SÁNCHEZ, 2015). Con el cálculo de $Ct$ se demuestra la existencia de un 100% de concordancia con la solución.

*Tabla 30: Coeficiente de concordancia total de la solución (Fuente: elaboración propia).*

| Votos negativos | Votos totales | Coeficiente de Concordancia total |
|---|---|---|
| $Vn$ | $Vt$ | $Ct = \left(1 - \frac{Vn}{Vt}\right) * 100$ |
| 0 | 50 | 100% |

Los resultados de la validación de la solución a través del criterio de expertos permite confirmar la aplicabilidad de esta y concluir que:

- El apoyo de la solución al proceso de migración de servicios telemáticos; la posibilidad que la solución seleccione el servidor web teniendo en cuenta el tipo de contenido a publicar y el total de peticiones concurrentes; y los 60 escenarios definidos para la selección del servidor, son tres aspectos evaluados de Muy Adecuado.
- La solución propuesta como vía para una correcta selección del servidor web a instalar; y como medio para aumentar la eficiencia en la selección de los servidores web Apache 2 y Nginx, son dos aspectos evaluados de Bastante Adecuado.

### 3.4 Satisfacción de los especialistas en servicios telemáticos

Se realizó una encuesta a 10 especialistas en servicios telemáticos pertenecientes al Centro de Software Libre de la UCI (ver Anexo 8), utilizando la técnica de Iadov, para determinar su grado de satisfacción con la herramienta propuesta a partir de su experiencia en la migración y administración de servicios telemáticos. La técnica de Iadov se basa en el análisis de un cuestionario que tiene una estructura interna determinada, la cual sigue las relaciones que se establecen entre tres preguntas cerradas (cuya relación el sujeto desconoce) y el análisis posterior de cinco preguntas abiertas. La relación entre las preguntas cerradas se establece a través del denominado “Cuadro lógico de Iadov” (ver Tabla 31), indicando la posición de cada persona en la escala de satisfacción.

*Tabla 31: Cuadro lógico de Iadov (Fuente: elaboración propia).*

| ¿Le satisface la solución para la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto? | ¿Considera usted que se deba continuar realizando el proceso de selección del servidor web sin una guía que permita indicar el más eficiente teniendo en cuenta las características de la entidad a migrar? | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | No | | | No sé | | | Sí | | |
| | ¿Utilizaría la solución para la selección de los servidores web Apache 2 y Nginx y así mejorar la eficiencia en la selección? | | | | | | | | |
| | Sí | No sé | No | Sí | No sé | No | Sí | No sé | No |
| Me gusta mucho | 1 | 2 | 6 | 2 | 2 | 6 | 6 | 6 | 6 |
| No me gusta mucho | 2 | 2 | 3 | 2 | 3 | 3 | 6 | 3 | 6 |
| Me da lo mismo | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Me disgusta más de lo que me gusta | 6 | 3 | 6 | 3 | 4 | 4 | 3 | 4 | 4 |
| No me gusta nada | 6 | 6 | 6 | 6 | 4 | 4 | 6 | 4 | 5 |
| No sé qué decir | 2 | 3 | 6 | 3 | 3 | 3 | 6 | 3 | 4 |

El número resultante de la interrelación de las tres preguntas indica la posición de cada sujeto en la escala de satisfacción. La escala de satisfacción es la siguiente: (1 = Clara satisfacción; 2 = Más satisfecho que insatisfecho; 3 = No definida; 4 = Más insatisfecho que satisfecho; 5 = Clara insatisfacción; 6 = Contradictoria). De los 10 especialistas encuestados, 7 respondieron clara satisfacción, 2 más satisfechos que insatisfechos y 1 indefinido. A continuación se calcula el índice de satisfacción grupal (ISG), donde A, B, C, D, E, representan el número de sujetos con índice individual 1; 2; 3 ó 6; 4; 5, respectivamente y donde N representa el número total de sujetos del grupo.

$$ISG = \frac{A(1) + B(0{,}5) + C(0) + D(-0{,}5) + E(-1)}{N}$$

$$ISG = \frac{7(1) + 2(0{,}5) + 1(0) + 0(-0{,}5) + 0(-1)}{10}$$

$$ISG = \frac{7 + 1 + 0 + 0 + 0}{10} = 0{,}8$$

Posteriormente se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y - 1, como se observa en la Figura 11.

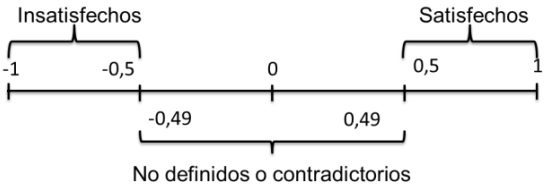


*Figura 11: Escala numérica de Iadov (Fuente: Sánchez, 2015).*

Debido a que el valor 0,8 se encuentra entre 0,5 y 1, indica que los especialistas en servicios telemáticos presentan satisfacción con la solución, resultante de la presente investigación.

### 3.5 Triangulación metodológica

La triangulación es un procedimiento de validez en el que los investigadores buscan la convergencia entre múltiples y diferentes fuentes de información para formar temas o categorías en un estudio. Por tanto, la triangulación es la combinación de dos o más enfoques metodológicos, perspectivas teóricas, fuentes de datos, investigadores y métodos de análisis para estudiar el mismo fenómeno. La triangulación metodológica se define como el uso de más de dos métodos para estudiar el mismo fenómeno bajo investigación. Se basa en el uso de métodos y análisis de recolección de datos tanto

cualitativos como cuantitativos en el estudio del mismo fenómeno (HUSSEIN, 2009). En la Tabla 32 se muestra el resultado de la triangulación metodológica de los métodos aplicados en la presente investigación.

*Tabla 32: Resultado de la triangulación metodológica (Fuente: elaboración propia).*

| Objetivo | Métodos cuantitativos | Métodos cualitativos | Conclusiones |
|---|---|---|---|
| Evaluar la eficiencia en la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto. | **Estudio de caso:** se compara la eficiencia en ECOAIND3 de dos proyectos, el primero sin aplicar la solución y el otro aplicándola. Se verifica que en el segundo la eficiencia es mayor que en el primero. | **Técnica de Iadov:** los especialistas en servicios telemáticos presentan satisfacción con la solución. $ISG = 0{,}8$<br>**Método Delphi:** de los cinco aspectos evaluados, tres se evaluaron de Muy Adecuado y dos de Bastante Adecuado, existiendo un 100% de concordancia con la solución. | Los resultados obtenidos en los métodos cuantitativos y cualitativos coinciden, por lo que se puede afirmar que la solución propuesta aumenta la eficiencia en la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto |

## 3.5 Conclusiones parciales

Al finalizar el presente capítulo se concluye lo siguiente:

- La realización de un estudio de caso en ECOAIND3 analizando dos proyectos de migración, permitió verificar el aumento de la eficiencia en la selección de los servidores web Apache 2 y Nginx, con la aplicación de la solución.
- El empleo del criterio de expertos mediante el método Delphi y el Modelo de Torgerson, permitió la valoración de cinco aspectos relacionados con la solución, resultando tres de ellos de Muy Adecuado (60%) y dos de Bastante Adecuado (40%), demostrándose la existencia de un 100% de concordancia con la solución.
- La aplicación de la técnica de Iadov para medir el grado de satisfacción con la solución, permitió determinar que los especialistas en servicios telemáticos están satisfechos con la misma.
- La realización de la triangulación metodológica permitió determinar que tanto los métodos cualitativos como cuantitativos aplicados para evaluar la solución, arrojaron el mismo resultado satisfactorio evidenciándose el cumplimiento del objetivo.

# CONCLUSIONES GENERALES

Al finalizar la presente investigación se concluye lo siguiente:

- El análisis de las fuentes bibliográficas relacionadas con los servidores web Apache 2 y Nginx, permitió incluir como alternativa para las instituciones cubanas la configuración de Nginx como proxy inverso de Apache 2 y determinar que el correcto funcionamiento del servidor web depende de la arquitectura del servidor, la cantidad de peticiones concurrentes y el tipo de contenido publicado.
- El empleo de un estudio de caso aplicando la Norma Cubana ISO/IEC 25023:2017 y calculando la eficiencia a partir del tipo de contenido y la cantidad de peticiones concurrentes, permitió el desarrollo de un componente Web en la Herramienta para la Migración y Administración de Servicios Telemáticos, que selecciona el servidor web más eficiente en cuanto a 60 escenarios definidos.
- La validación de la solución mediante un estudio de caso en ECOAIND3 analizando dos proyectos de migración; el empleo del criterio de expertos a través del método Delphi y el modelo de Torgerson; la aplicación de la técnica de Iadov para medir el grado de satisfacción con la solución; y la triangulación metodológica; permitió evidenciar el cumplimiento del objetivo de la presente investigación.

## RECOMENDACIONES

Se recomienda para la continuación de la presente investigación:

- Evaluar la eficiencia en la selección del servidor web, aplicando la solución desarrollada en instituciones que publiquen contenido dinámico con lenguaje de programación PHP o Pyhton.
- Añadir a la solución desarrollada las medidas de adecuación funcional, en este caso de usarían para evaluar el grado en que el servidor web proporciona funciones que satisfacen las necesidades de la institución.

## REFERENCIAS BIBLIOGRÁFICAS

**APACHE SOFTWARE FOUNDATION**. Multi-Processing Modules (MPMs). [En línea]. 2017. [Consultado el: 20 de octubre de 2017]. Disponible en: [http://httpd.apache.org/docs/2.4/mpm.html].

**CAMPOVERDE**, Ariel M.; **HERNÁNDEZ**, Dixys L.; **MAZÓN**, Bertha E. Cloud computing con herramientas open-source para Internet de las cosas. Maskana, 2015, 6 (Número Especial): p. 173-182.

**CACCIAGRANO**, Diletta; **CORRADINI**, Flavio. On synchronous and asynchronous communication paradigms. En: 7th Italian Conference, ICTCS. Theoretical Computer Science. Torino, Italy: Springer, 2001, p. 256-268.

**CHARTE**, Francisco. Programación GNU/Linux. España, Anaya Multimedia, 2003. 720 p.

**DE LA TORRE**, César, et al. Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0 (BETA). España, Krasis Consulting, 2010. 433 p.

**EMBARAK**, Ossama. Data Analysis and Visualization Using Python. Analyze Data to Create Visualizations for BI Systems. New York, Apress, 2018. 374 p.

**ESCOBAR**, Jazmine; **CUERVO**, Ángela. Validez de contenido y juicio de expertos: una aproximación a su utilización. Avances en medición, 2008, 6 (1): p. 27-36.

**ETECSA**. Sitios web y portales nacionales de interés informativo, cultural e investigativo. [En línea]. 2018. [Consultado el: 5 de diciembre de 2018]. Disponible en: [http://www.etecsa.cu/inicio/sitios_web_nacionales/].

**GALANIS**, P. The Delphi method. Archives of Hellenic Medicine, 2018, 35 (4): p. 564-570.

**GARCÍA**, Mª Elena; **LENA**, Francisco Javier. Aplicación del método delphi en el diseño de una investigación cuantitativa sobre el fenómeno FABLAB. Empiria - Revista de metodología de ciencias sociales, 2018, 2 (40): p. 129-166.

**GILSTER**, Ron. PC Hardware: A Beginner's Guide. India, Tata McGraw-Hill Education, 2001. 674 p.

**HERNÁNDEZ**, Adrian. Propuesta de arquitectura de despliegue de clústeres de servidores web de altas prestaciones. [En línea]. Twelfth LACCEI Latin American and Caribbean Conference for Engineering and Technology, 2014. [Consultado el: 16 de noviembre de 2017]. Disponible en: [https://www.researchgate.net/publication/29213716].

**HELMKE**, Matthew. Ubuntu Unleashed 2015 Edition. Londres, Pearson Education, 2015. 912 p.

**HÖBEL**, Valentin. Paquete Estadístico: ajuste del sistema con herramientas de diagnóstico. Linux magazine, 2013, (92): p. 50-53.

**HUSSEIN**, Ashatu. The use of triangulation in social sciences research: Can qualitative and quantitative methods be combined? Journal of comparative social work, 2009, 4 (1): p. 1-8.

**KABIR**, Mohammed. La Biblia del Servidor Apache 2. España, Anaya Multimedia, 2003. 845 p.

**KHOLODKOV**, Valery. Nginx Essentials. Birmingham, Packt Publishing Ltd, 2015. 151 p.

**KUNDA**, Douglas; **CHIHANA**, Sipiwe; **SINYINDA**, Muwanei. Web Server Performance of Apache and Nginx: A Systematic Literature Review. Computer Engineering and Intelligent Systems, 2017, 8 (2): p. 43-52.

**LEWIS**, John; **LOFTUS**, William; **TAHILIANI**, Mohit P. *Java software solutions: foundations of program design*. Eighth Edition. Londres, Pearson Education, 2009. 832 p.

**LIGUS**, Slawek. Effective Monitoring and Alerting. Sebastopol, O'Reilly Media, Inc., 2013. 164 p.

**MARTÍNEZ**, Piedad Cristina. El método de estudio de caso. Estrategia metodológica de la investigación científica. Revista Científica Pensamiento y Gestión, 2006, (20): p. 165-193.

**MATAM**, Sai; **JAIN,** Jagdeep. Pro Apache JMeter: Web Application Performance Testing. New York, Apress, 2017. 347 p.

**MATOTEK**, Dennis; **TURNBULL**, James; **LIEVERDINK**, Peter. Pro Linux System Administration: Learn to Build Systems for Your Business Using Free and Open Source Software. Second Edition. New York, Apress, 2017. 1008 p.

**MCWHORTER**, Rochell R.; **ELLINGER**, Andrea D. Qualitative case study research: An initial primer. En: Handbook of research on innovative techniques, trends, and analysis for optimized research methods. Texas: IGI Global, 2018, p. 185-201.

**MEJÍA**, José Teodoro; **GONZÁLES**, María Isabel; **ESPAÑA**, Angel Rafael. Programming Algorithms of load balancing with HA-Proxy in HTTP services. *Journal of Science and Research:* Revista Ciencia e Investigación, 2018, 3 (CITT2017): p. 100-105.

**MOLINA**, Rachel. Módulo para administrar el servidor web Nginx desde la Herramienta para la Migración y Administración de Servicios Telemáticos. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, 2017.

**NAM**, Van. Comparative Performance Evaluation of Web Servers. VNU Journal of Science: Computer Science and Communication Engineering, 2017, 31 (3): p. 28–34.

**NANDAL**, Vikas, et al. Performance Testing on Web-based Application using LoadRunner. Journal of Emerging Technologies and Innovative Research (JETIR), 2018, 5 (9): p. 37-42.

**NEDELCU**, Clément. Nginx HTTP Server. Third Edition. Birmingham, Packt Publishing, 2015. 318 p.

**NETCRAFT**. Most Reliable Hosting Company Sites in November 2018. [En línea]. 2018. [Consultado el: 6 de diciembre de 2018]. Disponible en: [https://news.netcraft.com/archives/2018/].

**OFICINA NACIONAL DE NORMALIZACIÓN**. NORMA CUBANA NC ISO/IEC 9126-1:2005. INGENIERÍA DE SOFTWARE—CALIDAD DEL PRODUCTO—PARTE 1: MODELO DE LA CALIDAD (ISO/IEC 9126-1:2001, IDT). 2005.

**OFICINA NACIONAL DE NORMALIZACIÓN**. NORMA CUBANA NC ISO/IEC 25023:2017. INGENIERÍA DE SOFTWARE Y SISTEMAS – REQUISITOS DE LA CALIDAD Y EVALUACIÓN DE SOFTWARE Y SISTEMAS (SQuaRE) – MEDICIÓN DE LA CALIDAD DEL PRODUCTO DE SOFTWARE Y SISTEMA. 2017.

**PALMA**, Nurisel. Módulo para la administración de los servidores web en HMAST. Tesis para optar por el título de Ingeniero en Ciencias Informáticas, Universidad de las Ciencias Informáticas, La Habana, 2013.

**PARTIDO COMUNISTA DE CUBA**. Documentos del 7mo Congreso del Partido. Lineamientos de la Política Económica y Social del Partido y la Revolución para el período 2016-2021. UEB Gráfica Villa Clara, 2017.

**PÉREZ**, Yoandy; **GARCÍA**, Abel; **GOÑI**, Angel. Buenas Prácticas para la Migración a Código Abierto. La Habana, Ediciones Fututo, 2015. 106 p.

**PÉREZ**, Yoandy. Estrategia para la migración a aplicaciones de código abierto. Tesis para optar por el título de Máster en Informática Aplicada, Universidad de las Ciencias Informáticas, La Habana, 2015.

**RAKHMAWATI**, Nur Aini, et al. A survey of web Technologies in Indonesia Local Governments. Jurnal Sisfo, 2018, 7 (3): p. 213-222.

**REAL ACADEMIA ESPAÑOLA**. Diccionario de la lengua española. [En línea]. 2018. [Consultado el: 10 de marzo de 2018]. Disponible en: [http://dle.rae.es].

**RODRÍGUEZ**, Lissy. Informatización de la sociedad: principios y resultados de una política. [En línea]. Periódico Granma, 13 de julio de 2017. [Consultado el: 15 de octubre de 2017]. Disponible en: [http://www.granma.cu/cuba/2017-07-13/informatizacion-de-la-sociedad-principios-y-resultados-de-una-politica-13-07-2017-14-07-49].

**HERNÁNDEZ**, Roberto; **FERNÁNDEZ**, Carlos; **BAPTISTA**, Pilar. Metodología de la investigación. Cuarta Edición. México, McGraw-Hill, 2006. 882 p.

**SÁNCHEZ**, Susana. Estrategia de soporte técnico para el proceso de migración a código abierto en los Organismos de la Administración Central del Estado. Tesis para optar por el título de Máster en Informática Aplicada, Universidad de las Ciencias Informáticas, La Habana, 2015.

**SCHIAVI**, Pablo. La protección de los datos personales en las redes sociales. A&C-Revista de Direito Administrativo & Constitucional, 2013, 13 (52): p. 145-178.

**SONI**, Rahul. Nginx: From Beginner to Pro. New York, Apress, 2016. 255 p.

**STATISTA**. Number of internet users worldwide from 2005 to 2017 (in millions). [En línea]. 2018a. [Consultado el: 13 de diciembre de 2018]. Disponible en: [https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/].

**STATISTA**. Number of internet users worldwide from 2009 to 2017, by region (in millions). [En línea]. 2018b. [Consultado el: 13 de diciembre de 2018]. Disponible en: [https://www.statista.com/statistics/265147/number-of-worldwide-internet-users-by-region/].

**STATISTA**. ¿Cuántos usuarios de Internet hay en América Latina? [En línea]. 2018c. [Consultado el: 13 de diciembre de 2018]. Disponible en: [https://es.statista.com/grafico/13903/cuantos-usuarios-de-internet-hay-en-america-latina/].

**UCI**. Portal de Nova. [En línea]. 2018a. [Consultado el: 18 de junio de 2018]. Disponible en: [https://www.nova.cu/es/contenido/acerca-de].

**UCI**. Portal de Nova. [En línea]. 2018b. [Consultado el: 18 de junio de 2018]. Disponible en: [https://www.nova.cu/es/productos/nova-servidores].

**VAN**, Tien; **KRIEGER**, Udo R.; **CHAKKA**, Ram. Performance modeling of an apache web server with a dynamic pool of service processes. Telecommunication Systems, 2008, 39 (2): p. 117-129.

**W3TECHS**. Usage of web servers for websites. [En línea]. 2018. [Consultado el: 6 de diciembre de 2018]. Disponible en: [https://w3techs.com/technologies/overview/web_server/all].

# ANEXOS

## Anexo 1. Sitios de interés en Cuba publicados por ETECSA.

*Tabla 33: Sitios de interés en Cuba (Fuente: elaboración propia).*

| Nombre y dirección del sitio web | Sistema operativo | Servidor web | Lenguaje de programación |
|---|---|---|---|
| **Sitios de interés cultural y de entretenimiento** | | | |
| Actualidad del deporte cubano<br>http://www.jit.cu/ | Windows Server | IIS | ASP.NET |
| Programación, eventos y otras informaciones sobre el Circo Nacional de Cuba<br>http://www.circonacionaldecuba.cu/ | - | Apache | - |
| Revista digital con informaciones de interés para la juventud cubana.<br>http://www.somosjovenes.cu/ | Red Hat | Apache | PHP |
| Información sobre la cartelera cultural del país.<br>http://www.lapapeleta.cult.cu/ | - | Nginx | - |
| Sitio de promoción del Museo Nacional de Bellas Artes.<br>http://www.bellasartes.cult.cu/ | Ubuntu | Apache | - |
| Directorio web de la cultura cubana<br>http://www.sitiosculturales.cult.cu/ | Ubuntu | Apache | - |
| Portal del video clip cubano.<br>https://www.loslucas.icrt.cu/ | - | Nginx | PHP |
| Portal del cine cubano.<br>http://www.cubacine.cult.cu/ | Debian | Apache | - |
| Sitio oficial de Casa de las Américas.<br>http://www.casa.cult.cu/ | - | - | - |
| Portal de la TV Cubana. Cartelera de los canales de televisión<br>http://www.tvcubana.icrt.cu/ | - | Nginx | PHP |
| Publicación de la Federación de Mujeres Cubanas (FMC).<br>http://www.mujeres.co.cu/ | - | - | PHP |
| Publicación digital adscrita a la Casa Editora Abril.<br>http://www.juventudtecnica.cu/ | Red Hat | Apache | PHP |
| Portal de videojuegos cubanos<br>http://www.videojuego.cu/ | - | Apache | PHP |

| | | | |
|---|---|---|---|
| Sitio de la Fábrica de Arte Cubano<br>http://www.fac.cu/ | Debian | Apache | - |
| Centro de Informática en la Cultura CUBARTE<br>http://www.cubarte.cult.cu/ | - | Nginx | - |
| **Sitios de interés informativo** | | | |
| Sitio del gobierno de Cuba<br>http://www.cubadebate.cu/ | - | - | PHP |
| Órgano del Comité Central del Partido Comunista de Cuba (PCC)<br>http://www.granma.cu/ | Debian | Apache | PHP |
| Órgano de la Unión de Jóvenes Comunistas (UJC)<br>http://www.juventudrebelde.cu/ | Debian | Apache | PHP |
| Órgano de la Central de Trabajadores de Cuba (CTC)<br>http://www.trabajadores.cu/ | Debian | Apache | PHP |
| Portal Cuba sí con informaciones de Cuba y el mundo<br>http://cubasi.cu/ | - | - | PHP |
| Agencia informativa latinoamericana<br>https://www.prensa-latina.cu/ | - | Nginx | - |
| Sitio oficial de ETECSA<br>http://www.etecsa.cu/ | - | - | - |
| Páginas Amarillas de ETECSA<br>http://www.pamarillas.cu/ | RedHat | Apache | PHP |
| Contenidos Unificados para Búsqueda Avanzada (C.U.B.A.)<br>https://www.redcuba.cu/ | - | Nginx | PHP |
| Aduana General de la República<br>http://www.aduana.gob.cu/ | Red Hat | Apache | PHP |
| Información sobre las oficinas de correos, sus códigos postales y los servicios de su organización.<br>http://www.correos.cu/ | - | Apache | PHP |
| Publicación de la Agencia de Información Nacional (AIN)<br>http://ofertas.cu/ | - | - | - |
| Navegador cubano de Mapas y gestor de Destinos.<br>https://www.andariego.cu/ | - | Nginx | PHP |
| Sitio oficial del Banco Central de Cuba<br>http://www.bc.gob.cu/ | - | Apache | PHP |

| | | | |
|---|---|---|---|
| Servicios entuMovil de la empresa Desoft<br>https://www.entumovil.cu/ | CentOS | Apache | Python |
| Revista del Centro de Información para la Prensa cubana.<br>http://www.cubahora.cu/ | - | Apache | PHP |
| Sitio del Instituto de Meteorología<br>http://www.insmet.cu | Windows Server | IIS | ASP.NET |
| **Sitios de interés educativo investigativo** | | | |
| Portal educativo del Ministerio de Educación.<br>http://www.cubaeduca.cu/ | Ubuntu | Apache | PHP |
| Repasador en línea<br>https://repasador.cubaeduca.cu/ | - | Apache | PHP |
| Enciclopedia colaborativa cubana<br>https://www.ecured.cu/ | - | Nginx | PHP |
| Portal de la red de Salud de Cuba<br>http://www.infomed.sld.cu/ | Debian | Apache | PHP |
| Portal del medioambiente en Cuba.<br>http://www.medioambiente.cu/ | - | Apache | PHP |
| Red cubana de la ciencia<br>http://www.redciencia.cu/ | - | - | PHP |
| Portal de Editorial Universitaria<br>http://eduniv.mes.edu.cu/ | SUSE | Apache | - |
| Portal de la Universidad de Ciencias Informáticas.<br>http://www.uci.cu/ | - | Nginx | PHP |

**Anexo 2. Entrevista realizada a los especialistas en servicios telemáticos para determinar características del proceso de migración de los servidores web.**

Teniendo en cuenta su experiencia en la migración a software libre responda:

Pregunta 1: ¿Ha migrado algún servidor web?

Pregunta 2: ¿Existe alguna guía de cómo seleccionar el servidor web o elementos a tener en cuenta?

Pregunta 3: ¿Qué tiempo demora estudiar la alternativa a seleccionar?

Pregunta 4: ¿Están definidos los aspectos a considerar de cuándo es recomendable para la empresa o no, dependiendo de las características de esta?

Pregunta 5: ¿En qué se basan para seleccionar el servidor web a instalar?

Pregunta 6: Diga una vez migrado, qué herramienta emplea para administrar el servidor y las razones por las que la seleccionó. En caso de no usar ninguna herramienta diga sus razones.

**Anexo 3. Operacionalización de la variable dependiente.**

*Tabla 34: Indicadores para la medición de la eficiencia*
*(Fuente: OFICINA NACIONAL DE NORMALIZACIÓN, 2017).*

| Dimensión | Indicador | Medición | |
|---|---|---|---|
| Rendimiento | Tiempo medio de conclusión de un trabajo | $X = \sum_{i=1}^{n} (B_i)/n$ | $B_i$ = Tiempo de completar el trabajo i<br>n = Número de mediciones |
| | Adecuación del tiempo de conclusión de un trabajo | $X = A/B$ | A = Tiempo medio de conclusión de un trabajo medido por el indicador anterior<br>B = Tiempo de respuesta especificado |
| | Rendimiento medio | $X = n\sum_{i=1}^{n} (A_i/B_i)/n$ | $A_i$= Número de trabajos completados durante la i-ésimo tiempo de observación<br>$B_i$= i-ésimo período de tiempo de observación<br>n = Número de observaciones |
| Utilización de los recursos | La media de utilización del procesador | $X = \sum_{i=1}^{n} (A_i/B_i)/n$ | $A_i$ = Tiempo de procesador realmente utilizado para ejecutar un conjunto dado de tareas en la observación i<br>$B_i$ = Tiempo de operación para realizar las tareas en la observación i<br>n = Número de observaciones |
| | La media de utilización de la memoria | $X = \sum_{i=1}^{n} (A_i/B_i)/n$ | $A_i$ = Tamaño de la memoria realmente utilizada para realizar un determinado conjunto de tareas para el procesamiento de i-ésima muestras<br>$B_i$ = Tamaño de memoria disponible para realizar las tareas durante el procesamiento de la i-ésima muestras<br>n = Número de muestras procesadas |
| | La media del uso de los dispositivos de entrada/salida (E/S) | $X = \sum_{i=1}^{n} (A_i/B_i)/n$ | $A_i$= Duración del tiempo ocupado de los dispositivos de E/S para realizar un conjunto dado de tareas para la i-ésima observación<br>$B_i$= Duración de las operaciones de E/S para realizar las tareas para la i-ésima observación<br>n = Número de observaciones |

| | | | |
|---|---|---|---|
| | Utilización del ancho de banda | $X = A/B$ | A = Ancho de banda de transmisión real medido a lo largo del tiempo para realizar un conjunto dado de tareas<br>B = Capacidad de ancho de banda disponible para realizar un determinado conjunto de tareas |
| Capacidad | Capacidad de procesamiento de transacciones | $X = A/B$ | A = Número de transacciones realizadas durante el tiempo de observación<br>B = Duración de la observación |
| | Capacidad de acceso de usuario | $X = \sum_{i=1}^{n} A_i/n$ | $A_i$ = Número máximo de usuarios que pueden acceder simultáneamente al sistema en la i-ésima observación<br>n = Número de observaciones |

**Anexo 4. Entrevista realizada a los administradores de servidores web.**

Estimado(a) administrador.

La presente entrevista tiene como objetivo conocer cómo se lleva a cabo actualmente el proceso de selección de los servidores web en su institución. Responda si es posible, las siguientes preguntas:

Pregunta 1: ¿Publican algún sitio web?

Pregunta 2: ¿Cuántos usuarios acceden al sitio?

Pregunta 3: ¿Cuántos usuarios acceden al sitio simultáneamente?

Pregunta 4: ¿El contenido del sitio es estático o dinámico?

Pregunta 5: ¿Cuáles son las prestaciones de la PC? En cuanto a:

a) RAM

b) CPU

c) Disco duro

Pregunta 6: ¿Cuál es el servidor web empleado?

Pregunta 7: ¿En qué se basaron para seleccionar ese servidor web y no otro?

Pregunta 8: Si el servidor web empleado es Apache 2 o Nginx:

a) ¿Por qué Apache 2 y no Nginx y viceversa?

b) ¿Ha tenido buenos resultados empleando ese servidor?

## Anexo 5. Gráficos para medir la eficiencia de los servidores web

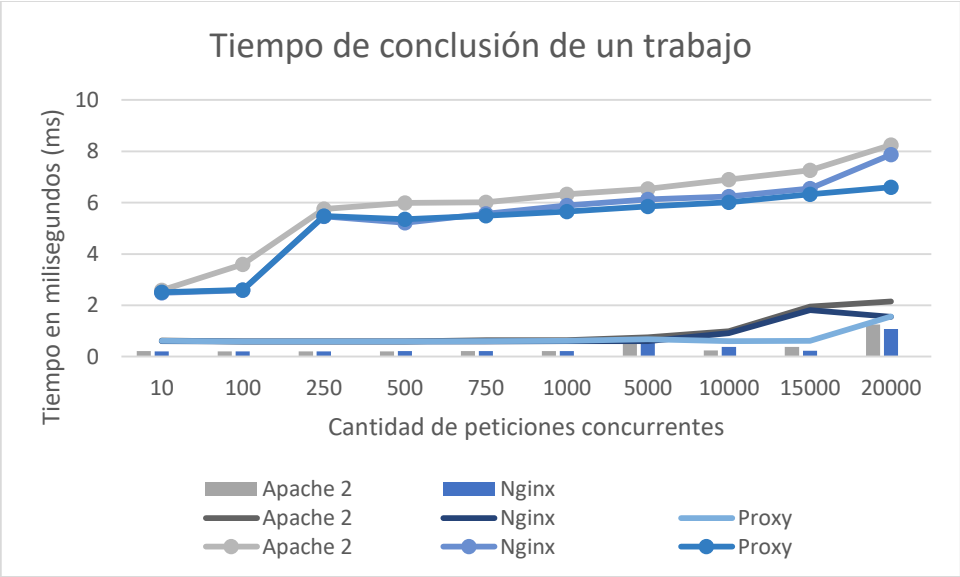


*Figura 12: Tiempo de conclusión de un trabajo (Fuente: elaboración propia).*

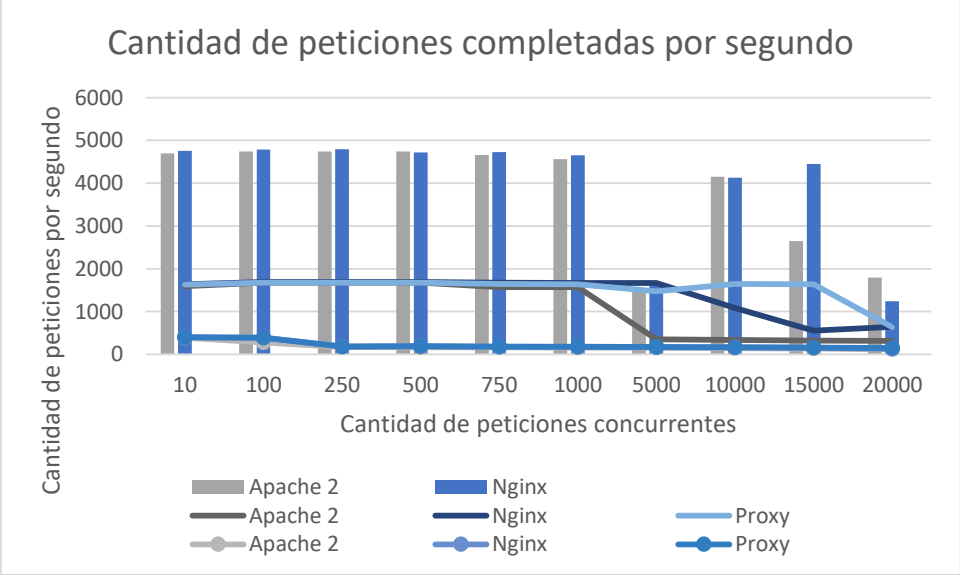


*Figura 13: Cantidad de peticiones completadas por segundo (Fuente: elaboración propia).*

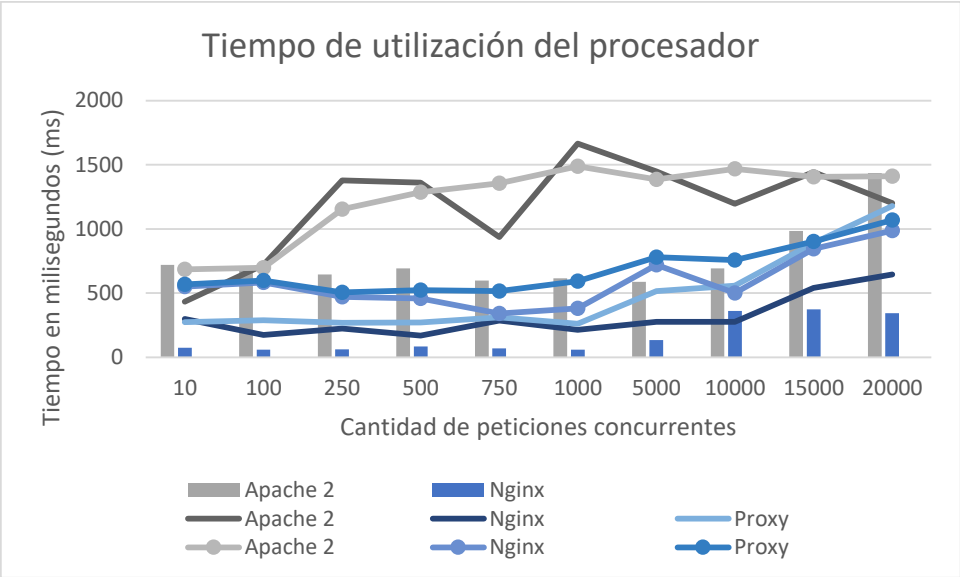


*Figura 14: Tiempo de utilización del procesador (Fuente: elaboración propia).*

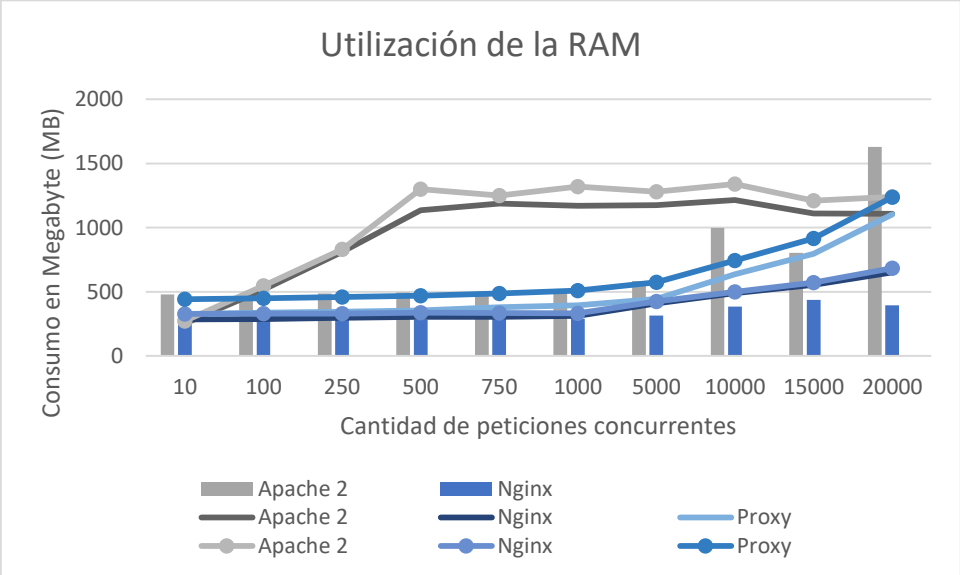


*Figura 15: Utilización de la RAM (Fuente: elaboración propia).*

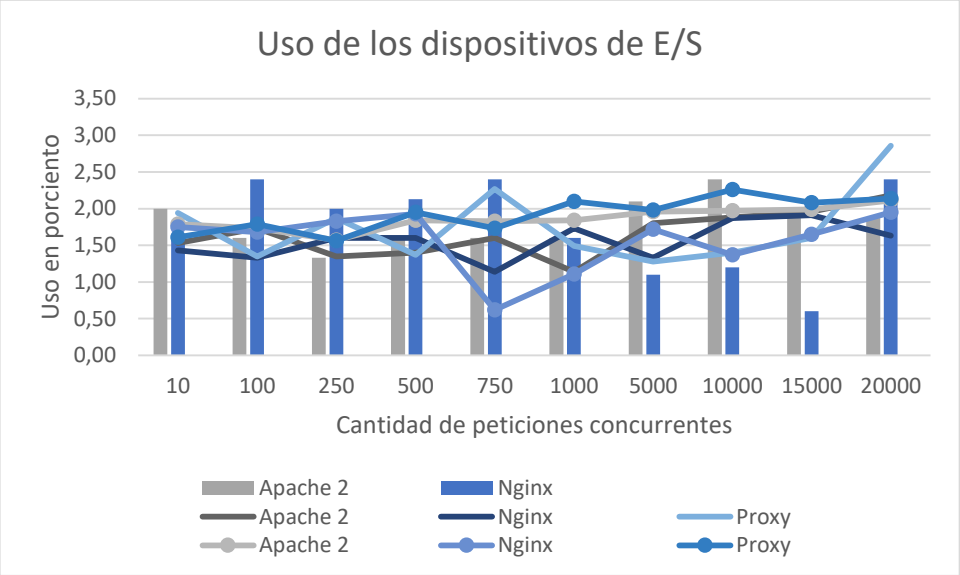


*Figura 16: Uso de los dispositivos de E/S (Fuente: elaboración propia).*

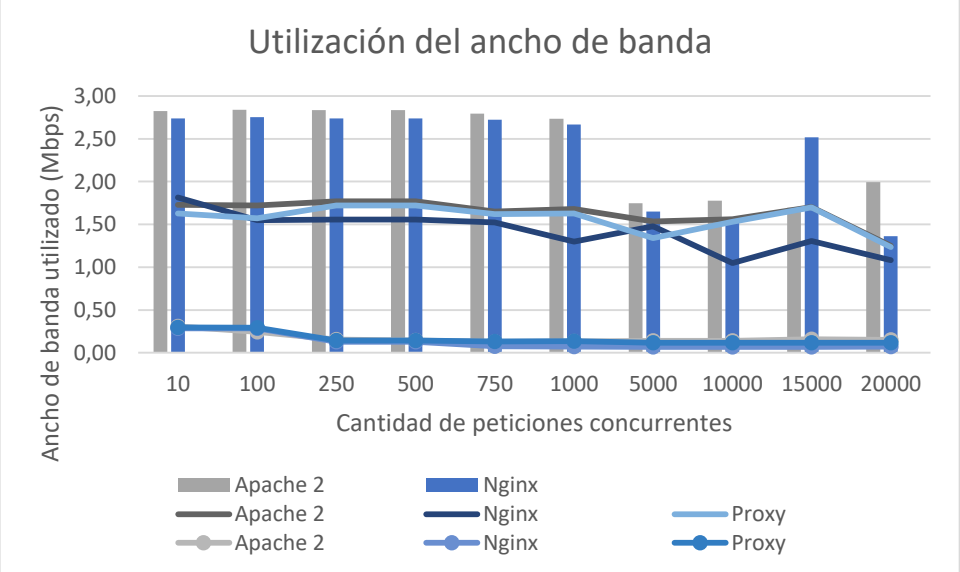


*Figura 17: Utilización del ancho de banda (Fuente: elaboración propia).*

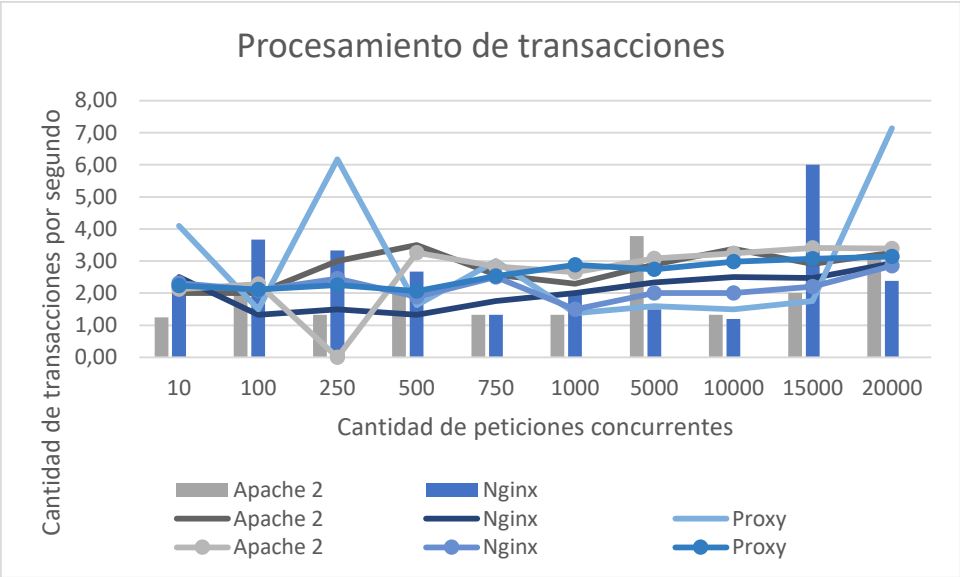


*Figura 18: Capacidad de procesamiento de transacciones (Fuente: elaboración propia).*

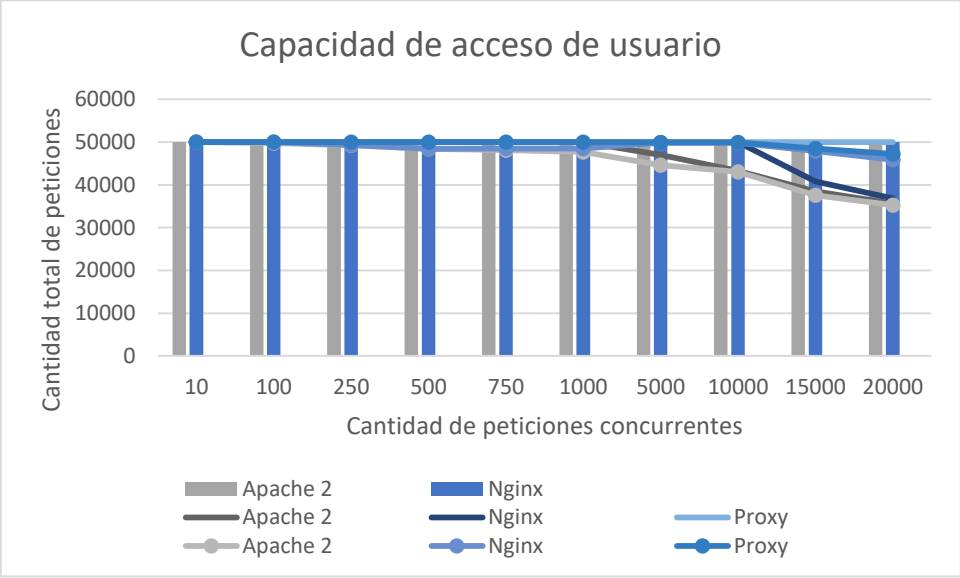


*Figura 19: Gráfica de la capacidad de acceso de usuario (Fuente: elaboración propia).*

**Anexo 6. Encuesta para determinar nivel de competencia de los expertos**

Estimado(a) compañero(a).

La encuesta que se presenta forma parte de las acciones para validar la solución para la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto. Le solicitamos la mayor responsabilidad y sinceridad en la realización de la encuesta. Se necesita primeramente que evalúe su conocimiento acerca de los servidores web y el grado de influencia de las fuentes de

argumentación, según las indicaciones que se dan a continuación. Le agradecemos de antemano por su valiosa contribución.

Datos generales del encuestado:

Título universitario: _____

Categoría científica: _____ Categoría docente: _____

Años de experiencia en el desarrollo de procesos de migración: _____

**Instrucciones**

1) Según su criterio, marque con una X en la casilla que caracteriza su nivel de conocimiento sobre los servidores web, “0” significa total desconocimiento del tema y “10” que tiene pleno conocimiento del mismo.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | |

2) Entre las fuentes que le han posibilitado enriquecer su conocimiento sobre el tema, se someten a consideración algunas de ellas, para que las evalúe en las categorías de: Alto (A), Medio (M) y Bajo (B), colocando una X.

| Fuentes de argumentación | Grado de influencia | | |
|---|---|---|---|
| | Alto (A) | Medio (M) | Bajo (B) |
| Estudios teóricos realizados por usted | | | |
| Experiencia adquirida durante su vida profesional | | | |
| Conocimiento de investigaciones y/o publicaciones nacionales e internacionales. | | | |
| Conocimiento propio sobre el estado del tema de investigación. | | | |
| Actualización en cursos de posgrado, diplomados, maestrías, doctorado. | | | |
| Intuición | | | |

MUCHAS GRACIAS POR SU COLABORACIÓN

**Anexo 7. Encuesta para determinar criterios de los expertos**

Estimado(a) compañero(a).

Con la finalidad de someter a su consideración como experto(a) la solución para la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto, se solicita su valoración sobre diferentes aspectos que a continuación se presentan. De antemano, le agradecemos su valiosa contribución. Para expresar su evaluación, por favor, luego de analizar cuidadosamente el

material que se adjunta, evalúe cada uno de los aspectos que se le presentan en la tabla, marcando con una cruz en la casilla correspondiente y teniendo en cuenta para ello el siguiente código de categorías de clasificación.

- **MA**: Muy adecuado.
- **BA**: Bastante adecuado.
- **A**: Adecuado.
- **PA**: Poco adecuado.
- **NA**: Nada Adecuado.

| No. | Aspectos | MA | BA | A | PA | I |
|---|---|---|---|---|---|---|
| 1 | La solución propuesta, como apoyo al proceso de migración de servicios telemáticos, la valora de … | | | | | |
| 2 | La posibilidad que la solución seleccione el servidor web teniendo en cuenta el tipo de contenido a publicar y el total de peticiones concurrentes, la valora de … | | | | | |
| 3 | Los 60 escenarios definidos para la selección del servidor, los valora de … | | | | | |
| 4 | La solución propuesta, como vía para una correcta selección del servidor web a instalar, la valora de ... | | | | | |
| 5 | La solución informática propuesta, como medio para aumentar la eficiencia en la selección de los servidores web Apache 2 y Nginx, lo valora de … | | | | | |

MUCHAS GRACIAS POR SU COLABORACIÓN

**Anexo 8. Encuesta para determinar nivel de satisfacción con la solución informática**

Estimado especialista en servicios telemáticos, la siguiente encuesta tiene el propósito de determinar su nivel de satisfacción en cuanto a la solución informática para la selección de los servidores web Apache 2 y Nginx.

1) ¿Considera usted que se deba continuar realizando el proceso de selección del servidor web sin una guía que permita indicar el más eficiente teniendo en cuenta las características de la entidad a migrar?

   _____ Sí _____ No _____ No sé

2) ¿Utilizaría la solución para la selección de los servidores web Apache 2 y Nginx y así mejorar la eficiencia en la selección?

   _____ Sí _____ No _____ No sé

3) ¿Observa diferencia entre el tiempo requerido para estudiar la alternativa a instalar en la institución y el tiempo que demora seleccionar el servidor web mediante la herramienta?

4) ¿Está de acuerdo con los elementos en los que se basa la solución informática para seleccionar el servidor web: cantidad de peticiones concurrentes y tipo de contenido?

5) ¿Qué opina acerca de los tiempos de respuesta del servidor web seleccionado?

6) ¿Qué opina acerca de la utilización de los recursos por el servidor web seleccionado?

7) ¿Qué opina acerca de la capacidad de acceso de usuario del servidor web seleccionado?

8) ¿Le satisface la solución para la selección de los servidores web Apache 2 y Nginx durante el proceso de migración a código abierto?

   ____ Me gusta mucho.

   ____ Me gusta más de lo que me disgusta.

   ____ Me da lo mismo.

   ____ Me disgusta más de lo que me gusta.

   ____ No me gusta nada

   ____ No sé qué decir