TRABAJO FINAL: BENCHMARK EN OS X

**Autores**: Javier Ramirez Quintero, José Jesús Torronteras Hernández y Juan José Mendez Torrero

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

12 de diciembre de 2016

Horas del trabajo: 50

índice

1. ESPECIFICAR LOS OBJETIVOS Y DEFINIR EL SISTEMA 3

1.1 MAMP. INSTALACIÓN, ANÁLISIS Y CONFIGURACIÓN 3

2. LISTAR LOS SERVICIOS QUE OFRECE EL SISTEMA Y SUS POSIBLES RESULTADOS 6

2.1 servidor http apache 6

2.2 Nginx 6

2.3 MySQL 7

2.4 php 7

2.5 python 7

2.6 perl 8

3. SELECCIONAR LAS MÉTRICAS 9

4. LISTAR LOS PARÁMETROs QUE PUEDEN AFECTAR A LAS PRESTACIONES 10

5. SELECCIONAR LAS TÉCNICAS DE EVALUACIÓN 11

6. SELECCIONAR LA CARGA DE TRABAJO 13

7. DISEÑAR LOS EXPERIMENTOS 14

8. ANALIZAR E INTERPRETAR LOS DATOS 15

10. CONCLUSIONES 27

11. PREGUNTAS 28

12. BIBLIOGRAFÍA 29

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

1. ESPECIFICAR LOS OBJETIVOS Y DEFINIR EL SISTEMA

El objetivo principal es un análisis de una base de datos implementada en nuestro sistema.

En nuestro caso, vamos a realizar un benchmark de las métricas que se nos han propuesto tanto acciones como inserciones, actualizaciones y borrado de datos en MySQL bajo Mac OS utilizando un servidor local.

En nuestro caso vamos a utilizar MAMP, se trata de una versión gratuita para uso personal, que nos permite ejecutar en un entorno local MySQL, Apache y PHP sin necesidad de tener que instalarlos por separado. Además nos proporciona la última versión de cada servicio.

MAMP viene con MySQL, que es el sistema de base de datos más comúnmente utilizado. Gracias a MAMP podemos desarrollar fácilmente aplicaciones de bases de datos MySQL complejas en nuestro sistema. Además MAMP incorpora phpMyAdmin lo que nos permite una fácil configuración de la base de datos.

Todo esto estará instalado bajo el sistema operativo Mac OS Sierra con las siguientes especificaciones:

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

* Procesador: Intel Core i5 de doble núcleo a 2,7 GHz
* RAM: 8GB de memoria LPDD3 a 1866 MHz
* SSD: 128 GB

1.1 MAMP. INSTALACIÓN, ANÁLISIS Y CONFIGURACIÓN

En la actualidad tenemos una gran variedad de programas que nos permite instalar un servidor local con servicios como Apache, Nginx, MySQL entre otros, destacando algunos como XAMPP para Mac OS, itools, TomCat, hubiC, etc.

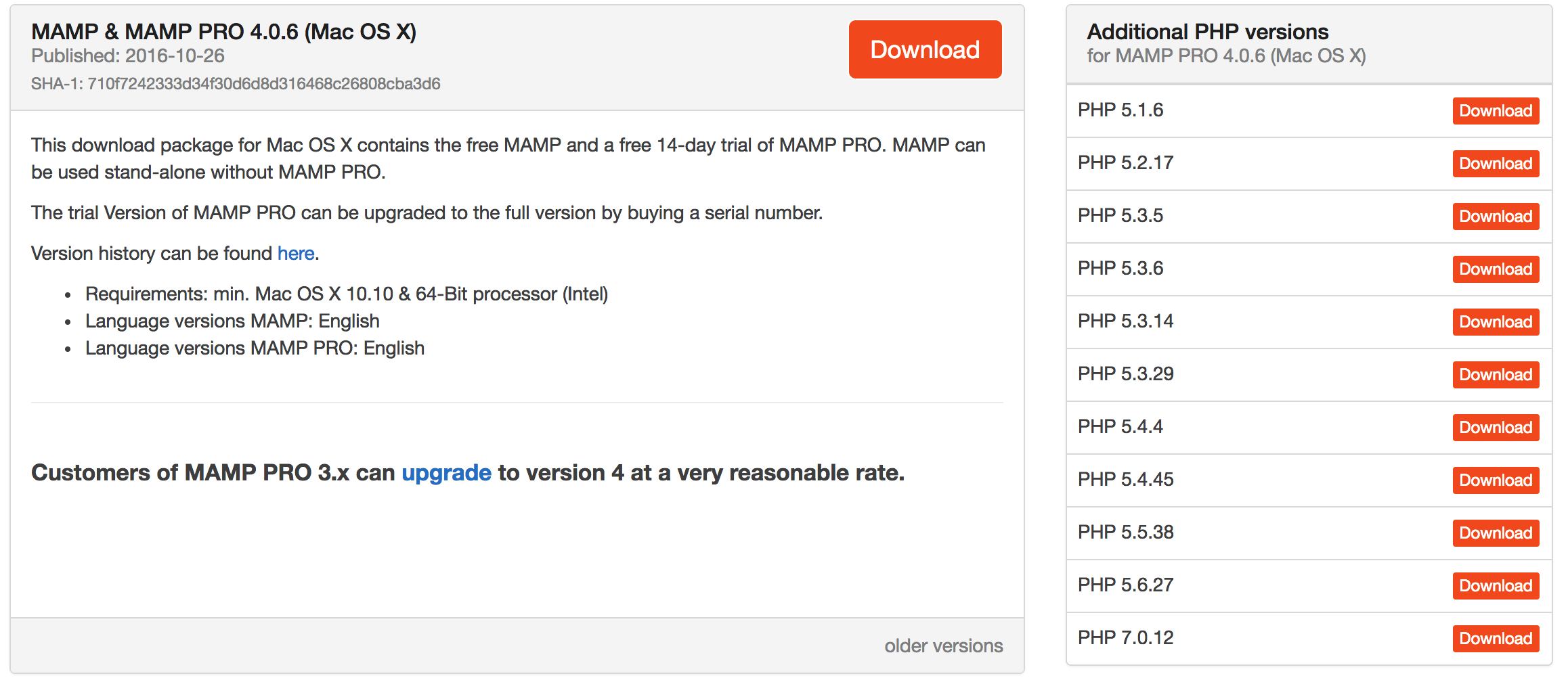


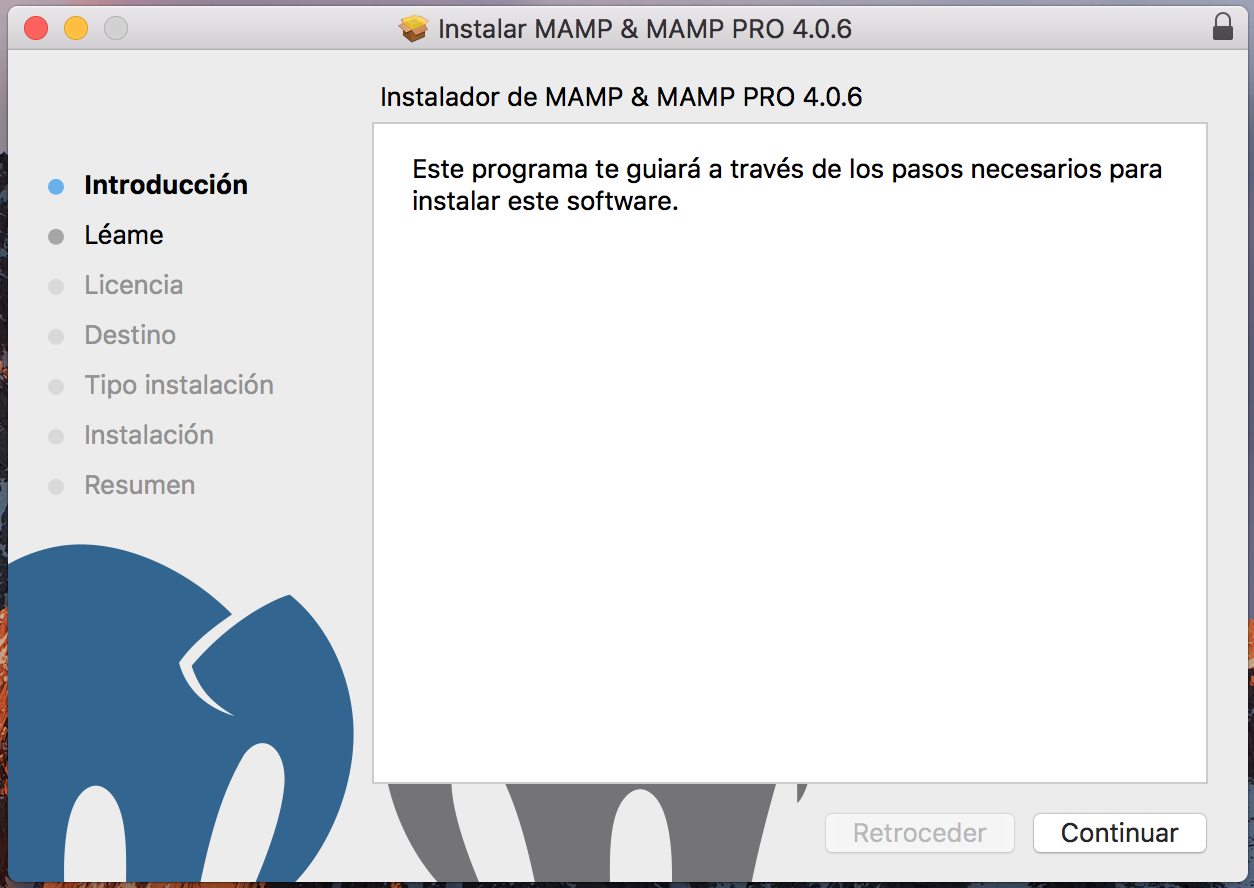
Hemos elegido MAMP (Mac + Apache + MySQL + PHP), ya que nos permite instalar un servidor local enfocado exclusivamente para Mac OS. Además destaca por su sencillez tanto de instalación, configuración y de su uso.

Por lo tanto MAMP nos ofrece todo lo necesario para poder realizar el benchmark de las acciones propuestas sobre la base de datos.

La instalación de MAMP es muy sencilla. Para descargarlo nos hemos dirigido a su página web oficial: <https://www.mamp.info/en/>

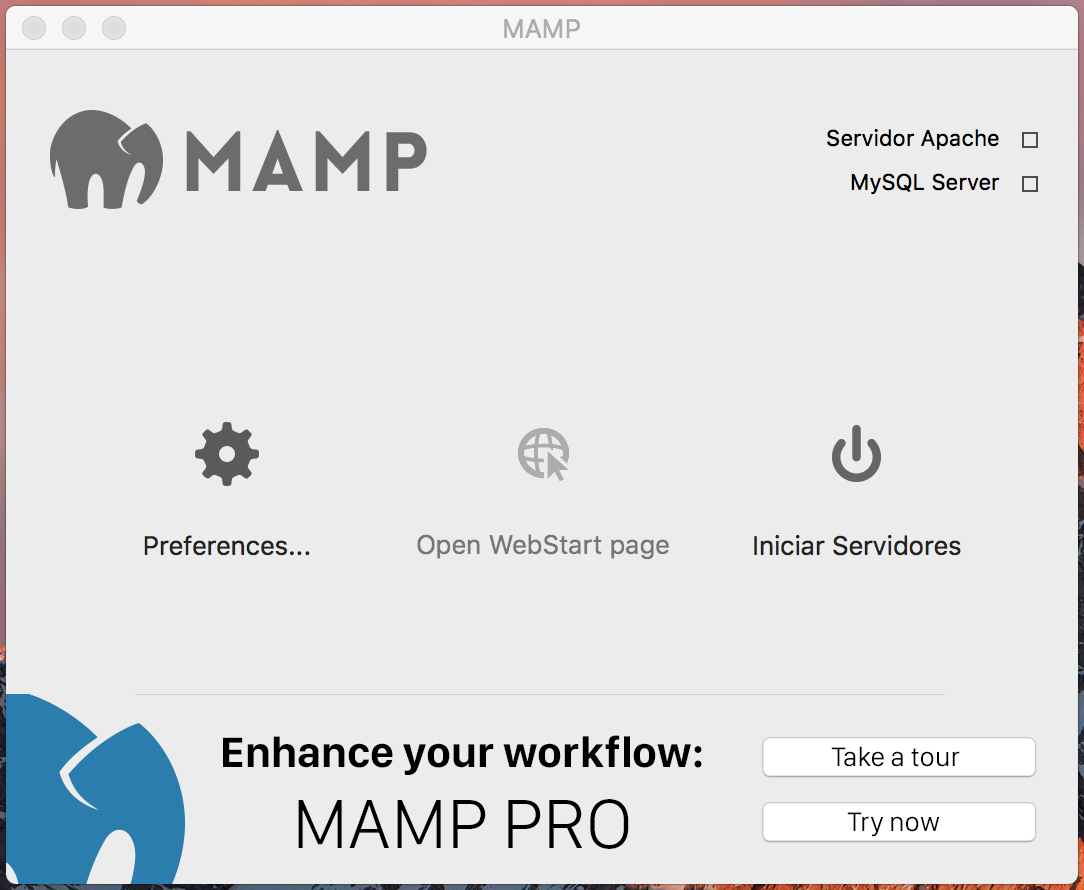
En la web encontramos infinidad de información sobre MAMP y de los servicios que nos ofrece, además cuenta con una amplia documentación.



Una vez descargado y ejecutado el paquete nos aparecerá el programa de instalación, donde solo tendremos que aceptar los términos y condiciones y establecer la ruta donde nos gustaría tener instalado MAMP. Esto es una gran ventaja frente a otros sistemas operativos ya que para instalar diversos programas tenemos que usar comandos o nos encontramos con instalaciones laboriosas.

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

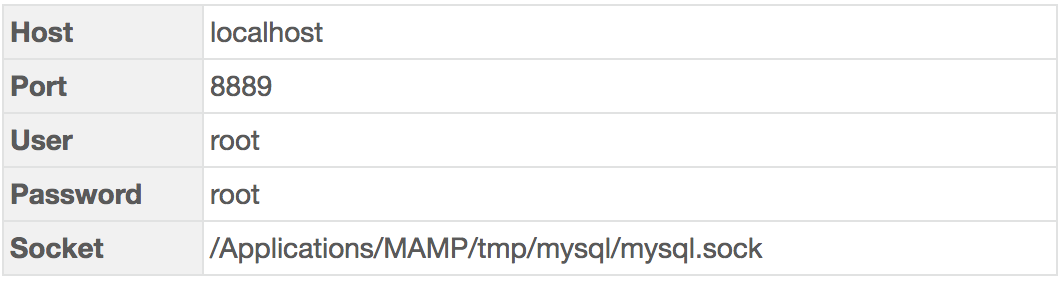
Observamos que su interfaz es realmente sencilla ya que lo único que tenemos son 3 botones que explicaremos a continuación e información sobre el servidor en la esquina superior derecha.



* Preferencias: Elemento principal del servidor, donde podremos configurar tanto puertos, como ver las versiones de los servicios que nos ofrecen.
* Iniciar Página Principal: Una vez iniciado los servidores tanto de Apache como MySQL, podemos tener un control de todo nuestro servidor gracias a esta página que nos ofrece MAMP. En ella podremos acceder a nuestra web en Localhost, obtener información sobre la versión del PHP instalado, acceder a herramientas como phpmyadmin para poder gestionar y acceder a nuestra base de datos y además cuenta con guías en Python y Perl.

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

* Iniciar Servidores: Este botón iniciará los servidores de Apache y MySQL necesarios en nuestro servidor.

Para configurar nuestro servidor estableceremos la versión de cada servicio y también configuraremos la base de datos estableciendo puertos, usuario y contraseña.

2. LISTAR LOS SERVICIOS QUE OFRECE EL SISTEMA Y SUS POSIBLES RESULTADOS

Como se ha expuesto anteriormente, MAMP instla un entorno de servidor local que incluye varios servicios que se expondrán en este apartado. Entre estos servicios se encuentran: Apache, Nginx, MySQL, PHP, Python y Perl.

2.1 servidor http apache

Este servicio que ofrece MAMP, es un servidor web HTTP de código abierto para plataformas UNIX, Windows, Mac OS, entre otras. Este servidor implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero es muy criticado por su falta de una interfaz gráfica que ayude en su configuración. Otra característica es que tiene una amplia aceptación en la red ya que ayudó al desarrollo de la Worl Wide Web. La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache. Las ventajas que presenta son, que es modular, es decir, facilita mucho la tarea de codificarlo, es de código abierto, con lo que cualquier programador puede ver su código. Además es multi-plataforma, ya que es soportado por la mayoría de SO, también es extensible y muy popular, junto con que es muy fácil conseguir ayuda por parte del soporte técnico.

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

<https://es.wikipedia.org/wiki/Servidor_HTTP_Apache>

2.2 Nginx



Es un servidor web ligero de alto rendimiento y un proxy para protocolos de correo electrónico. Además es un software libre y de código abierto, licenciado bajo la Licencia BSD simplificada. También es multiplataforma, por lo que corre en sistemas de tipo Unix y Windows. El sistema es usado por una larla lista de web conocidas, como: WordPress, Netflix, GitHub, etc.

Entre las características que presenta el servidor web se encuentran: Servidor de archivos estáticos, índices y autoindexado, balanceo de carga, tolerancia a fallos, soporte de HTTP y HTTP2 sobre SSL, es compatible con IPv6, también incluye un soporte para protocolo SPDY, compresión gzip, etc. Además, las características del proxy de correo son: Proxy SMTP, POP3 e IMAP y soporta STARTTLS y SSL.

<https://es.wikipedia.org/wiki/Nginx>

2.3 MySQL

Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base de datos open source más popular del mundo. Al contrario de proyectos como Apache, donde el software es dearrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

MySQL es muy utilizado en aplicaciones web, como Joomla WordPress o phpBB. Es una base de datos muy rápidad en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de entornos de alta concurrencia en la modificación. Funciona sobre múltiples plataformas, incluyendo AIX, BSD, Mac OS X, entre otras muchas.

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad y transacciones. Pero a pesar de ello, atrajo a los desarrolladores de páginas web con contenidoo dinámico. Poco a poco los elementos de los ue carecía están siendo incorporados tanto por desarrollos internos, como por desarolladores de software libre. Esntre las características se puede destacar: amplio subconjunto del lenguaje SQL, disponibilidad en gran cantidad de plataformas y sistemas, posiblidad de selección de mecanismos de almacenamiento que ofrecen diferentes velocidades de opreación, soporte físico, capacidad, etc. También realiza transacciones y claves foráneas, conectividad segura, búsqueda e indexación ded campos de texto. MySQL está escrito en una mezcla de C y C++.

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

<https://es.wikipedia.org/wiki/MySQL>

2.4 php

PHP es un lenguaje de programación de uso general de código del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se incorpora directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos.

Entre sus características destacan: Es orientado al desarrollo de aplicaciones web dinámicas, es un lenguaje fácil de aprender, el código fuente escrito en PHP es invisible al navegador web y al cliente, ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Además es libre, por lo que se presenta como una alternativa de fácil acceso para todos.

Como hemos visto, presenta muchas ventajas con respecto a otros lenguajes, pero también presenta inconvenientes, tales como: es un lenjuage que se interpreta en ejecución, debido a esto, un script en PHP suele funcionar considerablemente más lento que su equivalente en un lenguaje de bajo nivel, sin embargo este inconveniente se puede minimizar con técnicas dde caché tanto en archivos como en memoria.

<https://es.wikipedia.org/wiki/PHP#Caracter.C3.ADsticas_de_PHP>

2.5 python



Es un lenguaje de programación interpretado cuya filosofía hace incapié en una sintaxis que favorezca un código legible. Es administrado por la Python Software Foundation y posee una licencia de código abierto denominada Python Software License, que es compatible con la licencia pública general de GNU.

Entre sus características, cabe destacar que es un lenguaje de programación multiparadigma, esto significa que más que forzar a los programadores a adoptar un estilo particular de programación, permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Además usa tipado dinámico y conteo de referencias para la administración de memoria.

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

<https://es.wikipedia.org/wiki/Python#Caracter.C3.ADsticas_y_paradigmas>

2.6 perl

Perl es un lenguaje de propósito general originalmente desarrollado para la manipulación de texto y que ahora es utilizado para un amplio rango de tareas incluyendo administración de sistemas, desarrollo web, programación en red, desarrollo de GUI y más.

La estructura completa de Perl deriva ampliamente del lenguaje C. Es un lenguaje imperativo, con variables, expresiones, asignaciones, bloques de código delimitados por llaves, estructuras de control y subrutinas. También toma características de la programación shell. Soporta estructuras de datos complejas, además tiene un modelo de programación orientada a objetos.

<https://es.wikipedia.org/wiki/Perl#Descripci.C3.B3n>

3. SELECCIONAR LAS MÉTRICAS

En esta sección se van a detallar las métricas usadas para comparar las prestaciones de nuestra base de datos. Hemos querido usar estas métricas pues son las que más se tienen en cuenta y las más importantes a nivel de rendimiento a la hora de definir el rendimiento de un servicio.

Se va a proceder a detallar las medidas relacionadas con las métricas usadas para la inserción, actualización y borrado de datos. Son las siguientes:

* Tamaño de la carga (Kb): hace referencia al tamaño de los datos a procesar por la base de datos en MySQL. Es uno de los factores más influyentes a la hora de determinar el rendimiento del servicio.
* Tiempo (ms): como su nombre indica, hace referencia al tiempo que ha tardado la base de datos en ejecutar una acción. Es también otra de las métricas más importantes a analizar en nuestro trabajo.

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

* Lectura Disco (Kb/s): hace referencia a la velocidad con la que el sistema lee los archivos que se le solicitan del disco del sistema.
* Escritura Disco (Kb/s): hace referencia a la velocidad con la que el sistema escribe en el disco del sistema los archivos que se le solicitan.
* Uso de la CPU (%): hace referencia a la cantidad de CPU total del sistema que está siendo usada por el servicio en cada instante.
* Uso de memoria (%): hace referencia a la cantidad de memoria del sistema que está siendo usada por el servicio en cada instante.

4. LISTAR LOS PARÁMETROs QUE PUEDEN AFECTAR A LAS PRESTACIONES

Los parámetros que afectan a las prestaciones son dos: la carga de trabajo y el hardware de nuestro sistema.

Nuestro sistema, al no tratarse de una máquina virtual, las prestaciones del host son las mismas. A continuación, se procederá a detallar el hardware de nuestro sistema y cómo influye éste en las prestaciones:

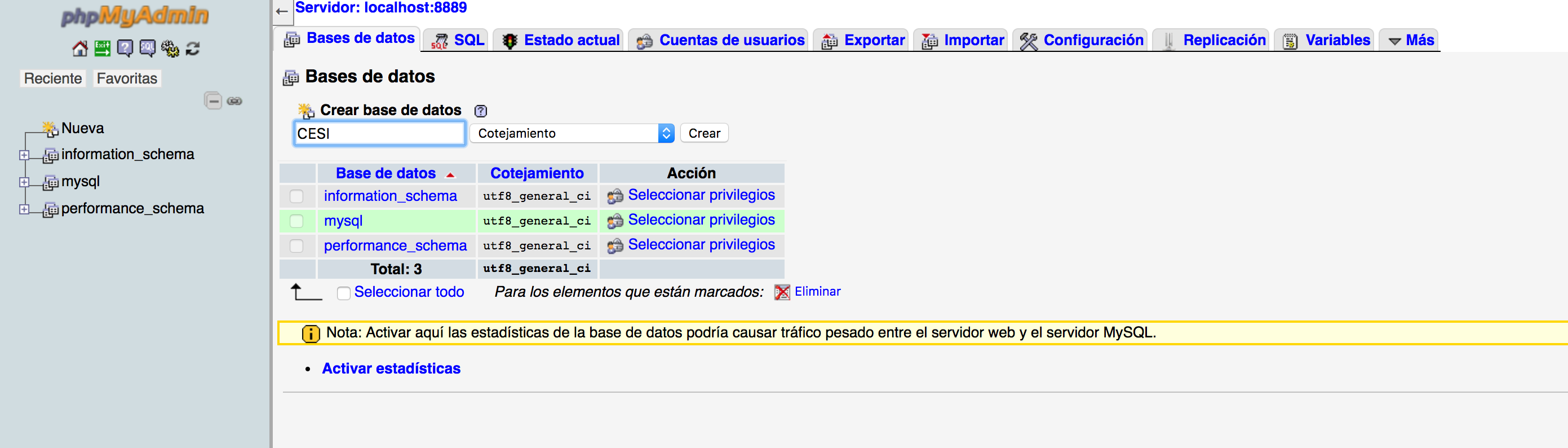
* CPU: parte fundamental del sistema el cual se encarga de interpretar la instrucción de los programas mediante la realización de operaciones. Nuestro trabajo se centrará sobre todo en este componente pues es el que rige el funcionamiento de los demás. La CPU consta de un procesador Intel Core i5 de doble núcleo a 2'7 GHz.

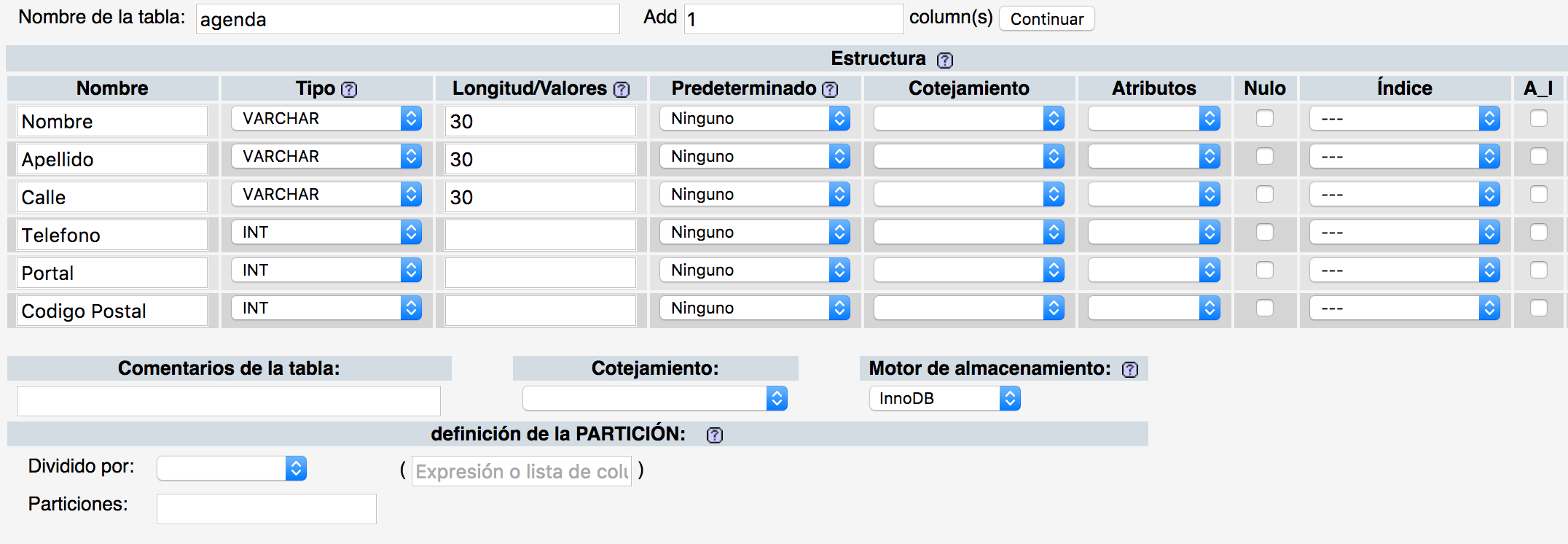
TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

* Memoria RAM: es otra de las partes fundamentales del sistema. En ella se cargan todas las instrucciones que se ejecutan en la CPU y en otras unidades. Nuestro sistema dispone de 8 Gb de RAM.
* Memoria Física: nuestro sistema dispone de 128 Gb, los cuales son más que suficientes para probar todos nuestros experimentos con la carga de trabajo definida.

El otro parámetro que afecta a nuestras prestaciones es la carga de trabajo, ésta va a definir la carga que va a tener que soportar los componentes hardware al ejecutar nuestra base de datos. La carga de trabajo y su contenido son definidos en el punto 7 de este trabajo.

5. SELECCIONAR LAS TÉCNICAS DE EVALUACIÓN

En esta sección se van a detallar las técnicas de evaluación llevadas a cabo para poder obtener los datos que necesitamos para el análisis del servidor MySQL. Lo primero que vamos a hacer es crear la base de datos, para ello, gracias a MAMP que cuenta con phpmyadmin, nos permite la creación de manera visual la tabla.

Primeramente, creamos la base de datos con el nombre de CESI, seguidamente hemos creado una tabla donde se almacenan los datos de una persona, y los campos podemos verlos en la siguiente imagen:

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

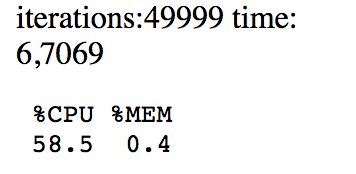
Para poder realizar la evaluación correctamente necesitamos que el servidor esté funcionando, para ello necesitamos un programa o script que se encargue tanto de la extracción de datos como de su análisis. En nuestro caso, hemos creado un script en PHP, donde el script se encargará de realizar la inserción/actualización/borrado de los datos en una tabla de MySQL creada manualmente por nosotros con PHPMyAdmin.

A continuación vamos a explicar nuestro script y la manera en que lo hemos codificado. Primero, conectamos el Script de PHP con la base de datos de MySQL para poder así realizar las acciones necesarias.

Seguidamente, con un bucle for, utilizamos la función mysqli\_query de la siguiente imagen para conectar con la base de datos y dependiendo de los comando que nosotros configuremos, inserta unos datos u otros.

Además, hemos encontrado una función que actúa como si fuera un cronómetro, para así poder saber el instante en el que los datos han sido insertados/borrados/actualizados. El cronómetro lo hemos codificado para que nos muestre el tiempo en que se ha hecho la acción, iteración por iteración.

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

Finalmente, para poder ver el rendimiento de nuestro PC, hemos utilizado una función en PHP denominada shell\_exec que nos permite ejecutar comandos en shell como por ejemplo, el comando ps, que es el que nosotros hemos utilizado para saber el rendimiento. El uso de esta función en PHP se muestra en la siguiente figura. Esto es una gran ventaja, ya que no necesitamos tener un terminal abierto para poder observar los resultados.

El comando ps nos permite visualizar el estado de un proceso, que si la utiliza con los opciones adecuadas, puedes seleccionar un proceso determinado, esto se consigue con la opción -p. Además, podemos, con la opción -o, seleccionar qué métricas queremos visualizar.

6. SELECCIONAR LA CARGA DE TRABAJO

La carga de trabajo es una parte fundamental para el desarrollo de la evaluación de nuestra base de datos, ya que es con lo que se define básicamente los objetivos de la base de datos.

Puede existir varios tipos de carga de trabajo, como número de consultas de datos simultáneamente, pero en nuestro caso nos hemos centrado en el rendimiento de inserción, actualización y borrado de datos de una base de datos en MySQL.

Para ello hemos definido una carga de prueba basada en un conjunto de tablas con diferentes cargas. Cada carga siempre utiliza la misma variable de registros

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

* Carga pequeña: Consiste en la inserción/borrado/actualización de 10.000 datos. Cuyo tamaño es de 1.5MB (1572864 Bytes).
* Carga mediana: Consiste en la inserción/borrado/actualización de 50.000 datos. Cuyo tamaño es de 3.5MB (3670016 Bytes).
* Carga grande: Consiste en la inserción/borrado/actualización de 100.000 datos. Cuyo tamaño es de 5.5MB (5767168 Bytes).

7. DISEÑAR LOS EXPERIMENTOS

En esta sección se va a proceder a detallar los experimentos realizador sobre el servidor MySQL, con los medios definidos en el apartado 6. Los experimentos irán destinados a analizar la inserción, actualización y borrado de los datos de nuestra tabla creada a través de MySQL. Todo esto se llevará a cabo mediante un script en PHP, en el cual vamos a poner una serie de funciones que se encargarán de estas acciones.

En cuanto a la inserción de datos, los hemos basado en el funcionamiento del comando mysqli\_query, y su estructura se muestra en la siguiente imagen:

Esta función tiene como parámetro principal un link de la base de datos que hemos conectado, y una consulta realizada a la base de datos ya preestablecida. En este caso, realizará una inserción dentro de la tabla agenda, ya creada previamente con los siguientes valores, que corresponden con los campos de la tabla. El experimento consiste en realizar un número elevado de iteraciones, y para ello utilizaremos un bucle for. Todo esto medirá todas las métricas que hemos declarado en el punto.

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

En lo referente al borrado, el funcionamiento del experimento es muy similar al realizado para la inserción de datos. Cambiando la consulta de la función mysqli\_query, conseguimos borrar los datos de la tabla agenda. En la siguiente figura se muestra cómo quedaría esta función para el borrado.

En cuanto a la modificación de los datos, es muy parecida a la de insertar, con la diferencia de que en este caso, los datos de la tabla agenda, son modificados campo a campo según se ponga dentro de la función mysqli\_query. En la siguiente figura vemos cómo se pueden cambiar, por ejempo, los campos de "Nombre" a "nombre" o de "Apellido" a "apellido".

8. ANALIZAR E INTERPRETAR LOS DATOS

Procedemos a hacer un benchmark del servidor MySQL, haciendo inserciones de datos:

| Análisis de inserción de datos en mysql | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Tamaño Carga (bytes) | Número de Iteracciones | Tiempo (segundos) | Lectura | Escritura | %CPU | %MEM |
| 1572864 | 1.000 | 0,69s | 16k | 528k | 39,7 | 0,3 |
| 3670016 | 5.000 | 1.81s | 0K | 108K | 67,3 | 0,3 |
| 5767168 | 10.000 | 4.39s | 0K | 756K | 58,8 | 4,4 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 2.000 | 1,08s | 28K | 820K | 57,1 | 0,3 |
| 3670016 | 10.000 | 3.68s | 12K | 316K | 64,8 | 0,3 |
| 5767168 | 20.000 | 8.6s | 0K | 856K | 64,7 | 4,4 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 3.000 | 1,47 | 28K | 872K | 60,9 | 0,3 |
| 3670016 | 15.000 | 5.37s | 24K | 784K | 65,4 | 0,3 |
| 5767168 | 30.000 | 12.5s | 0k | 768K | 63,10 | 4,4 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 4.000 | 2,21s | 24K | 764K | 42,1 | 0,3 |
| 3670016 | 20.000 | 7,3s | 32K | 780K | 62,8 | 0,3 |
| 5767168 | 40.000 | 16.4s | 0K | 816K | 63,7 | 4,4 |
|  |  |  |  |  |  |  |
| 1572864 | 5.000 | 3,56s | 24K | 788K | 31,2 | 0,3 |
| 3670016 | 25.000 | 8,96s | 24K | 748K | 66,6 | 0,3 |
| 5767168 | 50.000 | 20.5s. | 0K | 796K | 63,00 | 4,4 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 6.000 | 4,7s | 24K | 752K | 29,7 | 0,3 |
| 3670016 | 30.000 | 10,72s | 24K | 840K | 66,4 | 0,3 |
| 5767168 | 60.000 | 24.4s | 0K | 776K | 63,60 | 4,4 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 7.000 | 5,71s | 12K | 736K | 34,6 | 0,3 |
| 3670016 | 35.000 | 12,9s | 28K | 856K | 63,3 | 0,3 |
| 5767168 | 70.000 | 27.9s | 0K | 748K | 63,50 | 4,4 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 8.000 | 6,58s | 0K | 764K | 37,2 | 0,3 |
| 3670016 | 40.000 | 15,06s | 24K | 856K | 59,4 | 0,3 |
| 5767168 | 80.000 | 31.5s | 0K | 808K | 58,80 | 4,4 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 9.000 | 7,15s | 0K | 812K | 54,3 | 0,3 |
| 3670016 | 45.000 | 17,04s | 24K | 892K | 61,1 | 0,3 |
| 5767168 | 90.000 | 35.2s | 0K | 772K | 59,40 | 4,4 |
|  |  |  |  |  |  |  |
| 1572864 | 10.000 | 7,69s | 0K | 768K | 54,8 | 0,3 |
| 3670016 | 50.000 | 18,85s | 28K | 768K | 64,0 | 0,3 |
| 5767168 | 100.000 | 38.8s | 0K | 876K | 59,20 | 4,4 |

Para realizar el análisis nos hemos basado en las medias aritméticas de los datos, reflejadas en la siguiente tabla:

Como podemos observar el tiempo, va aumentando conforme se va aumentando el tamaño de la carga, esto se debe a que mientras más grande es el tamaño de los datos, mayor tiempo de inserción y mayor tiempo entre interrupciones de inserciones de nuevos datos

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

| Tamaño Carga (bytes) | Tiempo  (s) | Lectura  (Kb/s) | Escritura  (Kb/s) | CPU  % | MEM  % |
| --- | --- | --- | --- | --- | --- |
| Pequeña | 0,769 | 15,6 | 760,4 | 44,16 | 0,3 |
| Mediana | 1,885 | 22 | 694,8 | 64,11 | 0,3 |
| Grande | 3,85 | 0 | 797,2 | 61,78 | 4,4 |

También observamos como la escritura en disco se ve reducida para la carga mediana. En cuanto a la lectura de disco no se ve representada en ningún gráfico ya que en la inserción de datos no se ve afectada nada más que la escritura en disco.

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

Aquí observamos una comparativa del uso de la CPU y el de memoria. Como dato decir que en este benchmark hemos visto conveniente escoger el uso de CPU general en el sistema, cosa que no se verá reflejada en los siguientes benchmarks. Según nuestras observaciones mediante el benchmark, vemos que el uso de la CPU se mantiene estable y la memoria se ve incrementada un poco a mayor carga.

Seguidamente, procedemos a realizar el benchmark, pero esta vez, para la actualización de datos:

| Análisis de actualización de datos en mysql-1 | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Tamaño Carga (bytes) | Número de Iteracciones | Tiempo (segundos) | Lectura | Escritura | %CPU | %MEM |
| 1572864 | 1.000 | 107s | 0K | 140M | 30,7 | 0,6 |
| 3670016 | 5.000 | 168s | 0K | 133.2M | 32.5 | 0.7 |
| 5767168 | 10.000 | 250 | 0K | 106.7M | 36.1 | 1.2 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 2.000 | 215s | 0K | 113.8M | 96,1 | 0,6 |
| 3670016 | 10.000 | 312 | 0K | 131.2M | 94.2 | 0.7 |
| 5767168 | 20.000 | 361 | 0K | 128.2M | 98.1 | 1.2 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 3.000 | 284,08s | 0K | 121.1M | 95,3 | 0,6 |
| 3670016 | 15.000 | 360 | 0K | 129.5M | 96.4 | 0.7 |
| 5767168 | 30.000 | 421 | 0k | 131M | 93.2 | 1.2 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 4.000 | 357,35s | 0K | 109.3M | 84,7 | 0,6 |
| 3670016 | 20.000 | 450 | 0K | 127.4M | 85.1 | 0.7 |
| 5767168 | 40.000 | 513 | 0K | 129.1M | 87.6 | 1.2 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 5.000 | 432,36s | 0K | 105.3M | 96,2 | 0,6 |
| 3670016 | 25.000 | 520 | 0K | 123.4M | 97.5 | 0.7 |
| 5767168 | 50.000 | 604 | 0K | 144.3M | 98.1 | 1.2 |
|  |  |  |  |  |  |  |
| 1572864 | 6.000 | 505,162 | 0K | 116.3M | 92,2 | 0,6 |
| 3670016 | 30.000 | 610 | 0K | 129.9M | 91.5 | 0.7 |
| 5767168 | 60.000 | 697 | 0K | 132.9M | 94.2 | 1.2 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 7.000 | 577,072 | 0K | 113.8M | 97,3 | 0,6 |
| 3670016 | 35.000 | 701 | 0K | 129.9M | 94.9 | 0.7 |
| 5767168 | 70.000 | 761 | 0K | 153.9M | 98.1 | 1.2 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 8.000 | 649,25 | 0K | 105.3M | 92,5 | 0,6 |
| 3670016 | 40.000 | 794 | 0K | 114.5M | 90.1 | 0.7 |
| 5767168 | 80.000 | 839 | 0K | 149.8M | 93.2 | 1.2 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 9.000 | 719,14 | 0K | 116M | 94,3 | 0,6 |
| 3670016 | 45.000 | 831 | 0K | 120.9M | 98.6 | 0.7 |
| 5767168 | 90.000 | 940 | 0K | 106.7M | 93.5 | 1.2 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 10.000 | 789,9 | 0K | 132.9M | 94,0 | 0,6 |
| 3670016 | 50.000 | 911 | 0K | 128.6M | 96.4 | 0.7 |
| 5767168 | 100.000 | 1096 | 0K | 128.2M | 97.0 | 1.2 |

Para realizar el análisis nos hemos basado en las medias aritméticas de los datos, reflejadas en la siguiente tabla:

Como podemos observar el tiempo, va aumentando conforme se va aumentando el tamaño de la carga, esto se debe a que mientras más grande es el tamaño de los datos, mayor tiempo de actualización y mayor tiempo entre interrupciones de actualización de nuevos datos.

TRABAJO CONFIGURACIÓN Y evaluación de sistemas Informáticos

| Tamaño Carga (bytes) | Tiempo  (s) | Lectura  (Kb/s) | Escritura  (Mb/s) | CPU  % | MEM  % |
| --- | --- | --- | --- | --- | --- |
| Pequeña | 0,769 | 0 | 129.18 | 87.39 | 0.6 |
| Mediana | 1,885 | 0 | 141.83 | 96.59 | 0,7 |
| Grande | 3,85 | 0 | 131.08 | 88.91 | 3,4 |

Observamos como la escritura en disco se ve aumentada en la carga mediana. En cuanto a la lectura de disco no se ve representada en ningún gráfico ya que en la actualización de datos no se ve afectada nada más que la escritura en disco.

Procedemos a hacer un be2Wnchmark del servidor MySQL, pero esta vez, se hará un borrado de los datos de datos:

| Análisis de BORRADO de datos en mysql-2 | | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Tamaño Carga (bytes) | Número de Iteracciones | Tiempo (segundos) | Lectura | Escritura | %CPU | %MEM |
| 1572864 | 1.000 | 0.44 | 0K | 52K | 79.7 | 1.7 |
| 3670016 | 5.000 | 5.2 | 0K | 136,6M | 53.2 | 2.0 |
| 5767168 | 10.000 | 6.57 | 0K | 137.2M | 24.1 | 1.7 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 2.000 | 0.9 | 0K | 492K | 36.6 | 1.7 |
| 3670016 | 10.000 | 7.2 | 0K | 124,8M | 35.4 | 2.0 |
| 5767168 | 20.000 | 9.72 | 0K | 648K | 50.3 | 1.7 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 3.000 | 1.11 | 0K | 580K | 45.8 | 1.7 |
| 3670016 | 15.000 | 9.44 | 0K | 136,1M | 46 | 2.0 |
| 5767168 | 30.000 | 12.47 | 0K | 125.7M | 47.1 | 1.7 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 4.000 | 1.46 | 0K | 52K | 49.2 | 1.7 |
| 3670016 | 20.000 | 11.2 | 0K | 125,9M | 50 | 2.0 |
| 5767168 | 40.000 | 14.82 | 0K | 162.1M | 51.9 | 1.7 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 5.000 | 1.68 | 0K | 32K | 49.6 | 1.7 |
| 3670016 | 25.000 | 13.06 | 0K | 135,2M | 37.4 | 2.0 |
| 5767168 | 50.000 | 16.85 | 0K | 800K | 46.0 | 1.5 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 6.000 | 1.95 | 0K | 240K | 56.6 | 1.7 |
| 3670016 | 30.000 | 14.3 | 0K | 648K | 44.1 | 2.0 |
| 5767168 | 60.000 | 18.67 | 0K | 142.1M | 65.8 | 1.5 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 7.000 | 2.20 | 0K | 52K | 49.4 | 1.7 |
| 3670016 | 35.000 | 15.82 | 0K | 648K | 50.7 | 2.0 |
| 5767168 | 70.000 | 20.55 | 0K | 123.2M | 49.9 | 1.5 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 8.000 | 2.42 | 0K | 128K | 57.9 | 1.7 |
| 3670016 | 40.000 | 16.91 | 0K | 800K | 49.5 | 2.0 |
| 5767168 | 80.000 | 22.80 | 0K | 945K | 47.5 | 1.5 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 9.000 | 2.64 | 0K | 365K | 52 | 1.7 |
| 3670016 | 45.000 | 17.99 | 0K | 800K | 52.9 | 2.0 |
| 5767168 | 90.000 | 24.73 | 0K | 124.1M | 71.4 | 1.5 |
| Tamaño Carga (bytes) | **Número de Iteracciones** | **Tiempo (segundos)** | **Lectura** | **Escritura** | **%CPU** | **%MEM** |
| 1572864 | 10.000 | 2.94 | 0K | 35K | 50.7 | 1.7 |
| 3670016 | 50.000 | 19 | 0K | 728K | 58.6 | 2.0 |
| 5767168 | 100.000 | 26.8 | 0K | 848K | 49 | 1.5 |

Para realizar el análisis nos hemos basado en las medias aritméticas de los datos, reflejadas en la siguiente tabla:

Como podemos observar el tiempo, va aumentando conforme se va aumentando el tamaño de la carga, esto se debe a que mientras más grande es el tamaño de los datos, mayor tiempo de borrado y mayor tiempo entre interrupciones de borrado de nuevos datos.

| Tamaño Carga (bytes) | Tiempo  (s) | Lectura  (Kb/s) | Escritura  (Mb/s) | CPU  % | MEM  % |
| --- | --- | --- | --- | --- | --- |
| Pequeña | 2,94 | 0 | 2,028 | 52,75 | 1,7 |
| Mediana | 19 | 0 | 66,2224 | 47,78 | 2,0 |
| Grande | 26,8 | 0 | 81,7641 | 50,3 | 1,58 |

También observamos como la escritura en disco se ve aumentada en función al tamaño de las cargas. En cuanto a la lectura de disco no se ve representada en ningún gráfico ya que en la inserción de datos no se ve afectada nada más que la escritura en disco.

10. CONCLUSIONES

Tras la finalización del estudio realizado, podemos sacar una serie de conclusiones acerca del servidor MySQL. El objetivo principal de este estudio es medir las capacidades del sistema para poder obtener una conclusión de si es o no apto el servidor MySQL como servidor de archivos de uso específico. En primer lugar, según el análisis de los datos obtenidos a través de haber sometido al servidor a una carga, podemos afirmar que el servidor es apto para las tareas de insertar, borrar o modificar los datos. Aun así, podemos sacar diferentes conclusiones como:

* La inserción de datos en la tabla se hace a una velocidad mayor que la actualización, pero menor que el borrado.
* La actualización de los datos es la acción que más tarda en ejecutarse, además de ser la que más uso CPU utiliza.
* En cuanto al porcentaje de uso de memoria, como hemos podido ver en las tablas resumen que se ponen al final de cada tabla de iteraciones, la acción de inserción de datos con una carga grande, es la que más uso hace de ella.

11. PREGUNTAS

**1.¿Cuál es la desventaja de utilizar MAMP?**

La principal desventaja de utilizar MAMP es que nosotros estamos instalando servicios que ya están instalados predeterminadamente. Esto puede causar problemas cuando, por ejemplo, queremos utilizar PHP usando línea de comandos, esto activará el construito en la versión de PHP por defecto en lugar de la liada con MAMP. Si estos son diferentes, los resultados podrían ser inesperados.

**2.¿Qué es la Apache Software Foundation?**

Es una organización sin ánimo de lucro que fue creada para dar soporte a los proyectos de software bajo la denominación Apache, incluyendo el popular servidor HTTP Apache.

Es una comunidad de desarrolladores que trabajan de forma descentralizada y con los proyectos de código libre. Entre los objetivos de la ASF se encuentra el de proporcionar protección legal a los voluntarios que trabjan en proyectos Apache de ser empleado por otras organizaciones.

3.

12. BIBLIOGRAFÍA