Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Sistemas Operativos 1
Catedrático: Ing. Francisco Guevara Castillo.
Tutor Académico: José de Jesús Cano Rosales

Arquitectura Web Para Pruebas de Concurrencia

Proyecto de laboratorio

Objetivos Generales

- Implementar una arquitectura WEB con una aplicación media.
- Uso de herramientas para probar el rendimiento de la arquitectura.
- Comprender los problemas reales de concurrencia y las tecnologías y metodologías para solucionarlos.

Objetivos específicos

- Desarrollar una estructura modular que responda eficientemente a peticiones concurrentes
- Obtener métricas de concurrencia para realizar comparaciones en tiempos de respuesta.

Descripción

Se requiere la implementación de un servidor cloud, el cual deberá proveer una infraestructura como servicio (IaaS), que estará distribuida en varias máquinas virtuales a través de una plataforma de cloud como lo es google cloud platform. Esta infraestructura será capaz de almacenar los datos, llevando un registro de las peticiones web que han sido recibidas.

La finalidad de la red de servidores será mantener una alta disponibilidad, así como también comprobar la eficacia al verificar que la infraestructura soporta cargas masivas de tráfico de red y el almacenamiento de los datos.

IAAS

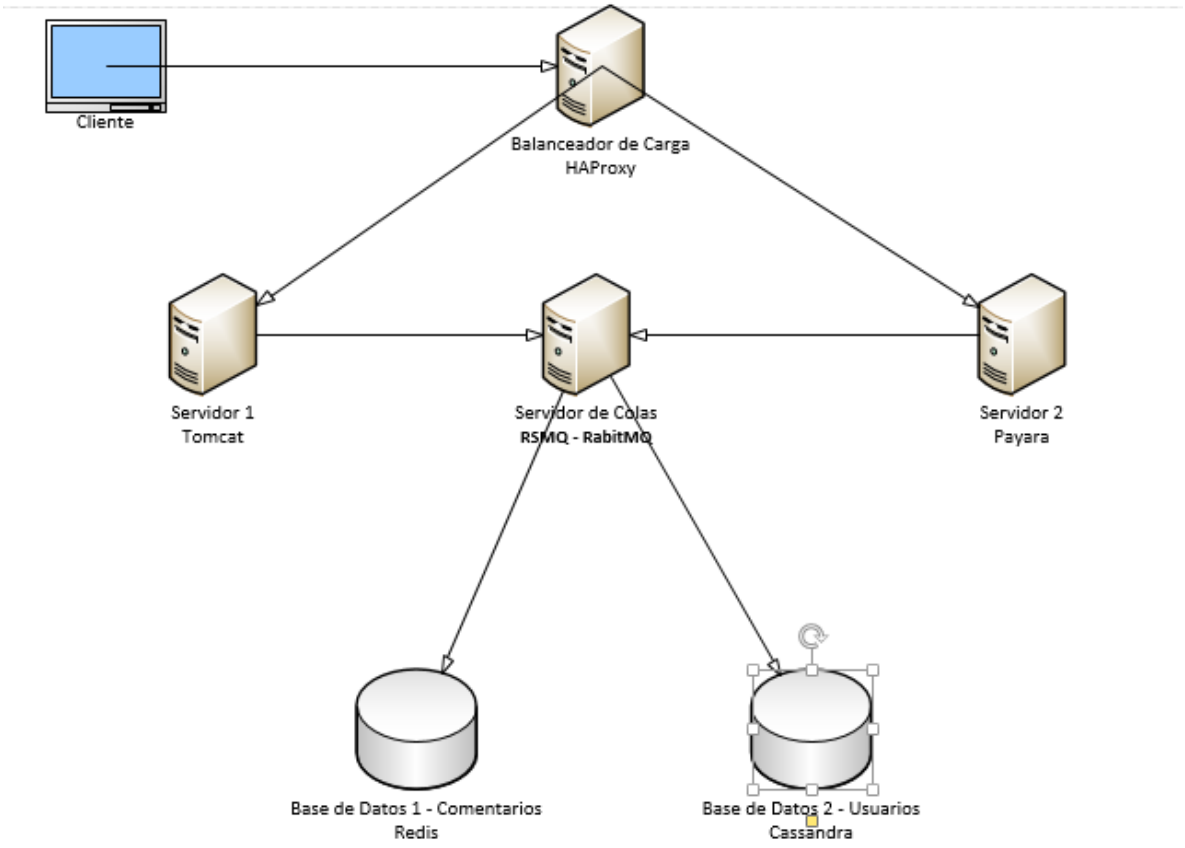
Se implementará un servidor de IaaS, en este caso, Google Cloud Platform será el encargado de proveernos este servicio. Esta tendrá alojada los servidores, cada uno aislado de los otros en una máquina virtual independiente. Se utilizará el sistema operativo Linux a elección más adelante descritos, para que los servidores sean lo más rápidos posible y que el consumo de recursos sea eficiente.

Aplicación

El proyecto consiste en crear una infraestructura básica para una red de mensajería, la cual llevara un control usuario o login, crear nuevo usuario, editar usuario y envió de mensajes la cual serán visualizado por todos los usuarios que tengan una cuenta.

Una de las partes más importantes de un sistema es la autenticación de sus usuarios. Es una de las partes básicas de la mayoría de los sistemas y esta debe de estar bien implementada para que no existan ingresos no autorizados. La práctica consistirá en el desarrollo de una arquitectura modular, la cual registrará controlará y verificará la identificación de usuarios.

Arquitectura



La distribución GNU/Linux recomendada es Ubuntu. Las otras distribuciones permitidas como servidores serán:
Fedora o Debian.

Cliente

El cliente es el que podrá visualizar las paginas, ingresando al servidor del HaProxy

LB(HaProxy)

Este servidor es un load balancer o balanceador de carga, debe de ser configurado con Haproxy este debera de recibir el trabajo y distribuirlo a los servidores Payara y Tomcat usando el algoritmo Round Robin.

Servidor 1 - Tomcat: Este servidor será el que tendrá una instancia de la aplicación. Sera una máquina virtual con un servidor Tomcat. Este estará contenido en una máquina virtual.

Servidor 2 - Payara: Este servidor será el que tendrá una segunda instancia de la aplicación. Sera una máquina virtual con un servidor **Payara**. El modularidad de este

proyecto recae en la reusabilidad de la aplicación, es decir, una vez hecha la aplicación solo será necesaria la instalación en otro servidor para que funcione de manera correcta. Este servidor estará contenido en una máquina virtual.

Servidor de cola de mensaje: este servidor contendrá una cola de mensajes, la cual se comunicará con la base de datos a utilizar (Redis) para procesar la lectura y escritura de datos y también contendrá otra cola de mensajes, la cual se comunicará con la base de datos (Cassandra), Es decir existirán dos colas de mensajes.

Se utilizará Rabbit como servidor de colas, verificar la documentación de Rabbit para ser el más adecuado a utilizar, se deberá de especificar en la documentación que modelo se utilizó para la implementación de colas.

Base de datos NoSql: esta contendrá la información registrada al sistema. Redis y Cassandra, pueden estar contenidas en la misma máquina virtual en donde se encuentra el servidor de colas de mensajes si se desea.

Redis: base de datos, esta base de datos almacenara lo siguiente

- Usuario
- Comentario realizado por el mismo usuario.

Cassandra: Base de datos para el control de usuarios, La aplicación a realizar deberá de ser un sistema de autenticación (Login) así como el registro de usuarios con un formulario de registro. Además, se deberá de poder ingresar a una interfaz para la consulta de usuarios registrados, siempre y cuando el usuario este autenticado. También se podrá visualizar la información de usuarios registrados por medio de una búsqueda.

De cada usuario se deberá de registrar:

- Nombre de usuario
- Contraseña
- Nombre
- Apellido
- Fecha de registro (con hora, minutos y segundos)

Se deberán de agregar las verificaciones necesarias para que se asegure que la contraseña es suficientemente fuerte, así como que el usuario este seguro de la contraseña a utilizar. Para garantizar la seguridad de estos datos, la contraseña no deberá de ser guardada como texto plano en el sistema, es decir no deberá de registrarse simplemente el campo con la contraseña ingresada, en su lugar lo que se guardará será la encriptación de la contraseña con SHA256.

Esta aplicación será una aplicación web, implementada en el lenguaje que se considere adecuado al fin del proyecto.

La arquitectura será puesta a prueba con la herramienta **Apache HTTP server benchmarking tool.** Las pruebas serán hacia cada uno de los servidores con la aplicación, así como al servidor del balanceo de carga. La parte a probar en esta prueba será una petición GET para obtener el número de usuarios registrados, así como una petición POST para registrar un numero aleatorio en la base de datos. Los resultados de las pruebas serán entregados en un documento comparativo para cada uno de los servidores probados con los siguientes parámetros:

- Concurrencia: 10; Peticiones 100
- Concurrencia: 10; Peticiones 1,000
- Concurrencia: 10; Peticiones 10,000
- Concurrencia: 100; Peticiones 10,000
- Concurrencia: 100; Peticiones 100,000
- Concurrencia: 100; Peticiones 1,000,000

Se deberá de concluir:

- De los 2 servidores de aplicaciones, cual es el que maneja mejor la concurrencia.
- La arquitectura funcionaria mejor con un solo servidor, o en su modalidad con un balanceador de aplicaciones.
- Si se pudo realizar todas las concurrencias solicitadas.

Restricciones:

- La aplicación web a desarrollar podrá ser realizada en cualquier framework o lenguaje de programación, siempre y cuando acepte las peticiones por HTTP para las pruebas a realizar (Peticiones GET y POST).
- El proyecto se realizará en parejas (2 personas).
- Se deberá tener un mínimo de 2 host para la calificación.
- Las tecnologías descritas en el enunciado son obligatorias (Base de datos, servidor de colas, herramienta de pruebas de carga, plataforma de virtualización y servidores a utilizar).
- **Por facilidad se recomienda hacer uso de google cloud para la realización del proyecto de igual tienen permitido realizar el proyecto en un ambiente local, haciendo uso de Ubuntu server (si se hace local se recomienda hacer uso de 512 mb a 1 Gb de ram para la virtualización para que no consuma demasiados recursos de la maquina física).**
- **Se deberá de realizar un script en bash para realizar concurrencia al servidor, haciendo únicamente solicitud el número de peticiones a realizar.**
- Se permite usar otras herramientas en sustitución de Tomcat y Payara, con una pequeña penalización en la calificación (Tomcat por Nginx y Payara por Apache).

Nota: Se dará puntos extras si se realiza el servidor de colas en **RSMQ**.

Entregables

- Aplicación funcional.
- Documenta técnica.
- Código de la aplicación.
- Script .sh

Fecha de entrega

La entrega se realizará subiendo el archivo a un link de Dropbox a más tardar el día martes 3 de julio del 2018 antes de las 11:00 PM horas en un archivo .zip con el nombre [SO1]Proyecto_#Grupo, La calificación de esta práctica se realizará el día miércoles 4 de julio.

Link: https://www.dropbox.com/request/MoQdKVphy7f4GawtjkGw

*Copias parciales o Completas serán ponderadas con nota 0.