Escenario y Pruebas de Estrés API REST y Batch

Integrantes

- Camilo Ramírez Restrepo
- Leidy Viviana Osorio Jiménez
- Laura Daniela Molina Villar
- Tim Ulf Pambor
- Shadith Perez Rivera

Objetivo

El objetivo de este plan es evaluar la capacidad de la aplicación Cloud conversión tool y su infraestructura de soporte en un entorno tradicional, para determinar sus máximos aceptables. El objetivo es comprender cómo la aplicación responde a diferentes niveles de carga de usuarios y cuál es su capacidad máxima.

Objetivos específicos

- Medir la capacidad de procesamiento de la aplicación en términos de transacciones por minuto.
- Evaluar el tiempo de respuesta promedio de la aplicación bajo diferentes cargas de usuarios.
- Analizar la utilización de recursos, como CPU, memoria y red, durante las pruebas de carga.
- Identificar cuellos de botella y posibles áreas de mejora en la infraestructura y la aplicación.

Descripción general

Aplicación web que ofrece gratuitamente a usuarios de internet subir abiertamente diferentes formatos multimedia de archivos y cambiarles su formato o realizar procesos de compresión. El modelo general de funcionamiento de la aplicación se basa en crear una cuenta en el portal web y acceder al administrador de archivos. Una vez la cuenta ha sido creada, el usuario puede subir archivos y solicitar el cambio de formato de estos para descargarlos. La aplicación web le permite a un usuario convertir archivos multimedia en línea de un formato a otro, seleccionando únicamente el formato destino.

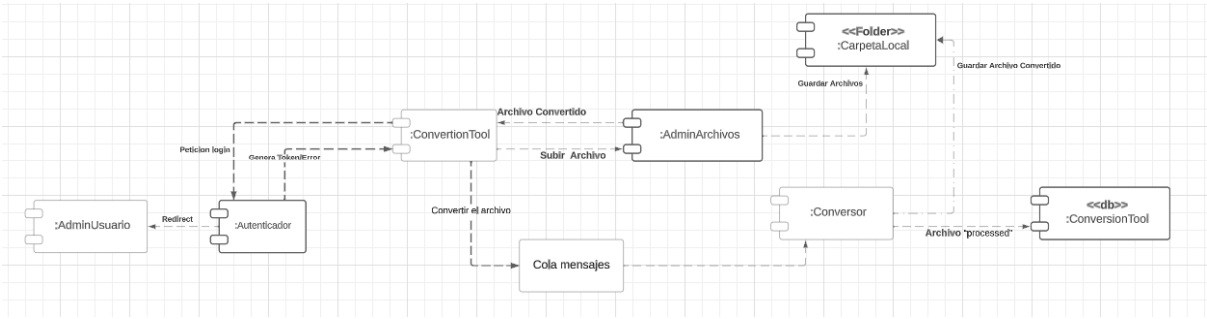

Imagen 1: Diagrama de información

Tipos de pruebas a realizar

- **Pruebas de carga:**

  Las pruebas de carga se realizan para evaluar cómo se comporta la aplicación bajo una carga de usuarios típica. Se simulan múltiples usuarios simultáneos realizando acciones en la aplicación, como navegación, búsqueda, compras, etc. El objetivo es medir el tiempo de respuesta promedio y asegurarse de que la aplicación funcione dentro de los límites aceptables bajo una carga normal.

- **Pruebas de estrés:**

  Las pruebas de estrés se utilizan para determinar el punto de quiebre de la aplicación y cómo responde bajo una carga excesiva. Se aumenta gradualmente la carga de usuarios hasta que se alcanza o supera el límite de capacidad de la aplicación. El objetivo es identificar cómo se comporta la aplicación bajo condiciones extremas y si es capaz de recuperarse sin fallos graves.

Configuración del sistema

- **Aplicación Flask**
- **Servidores EC2:**
  - Tipo de Instancia: Se utilizarán instancias de tipo t2.medium
  - Capacidad de CPU: Cada instancia t2.medium proporciona 4 vCPUs, lo que permite ejecutar múltiples instancias en paralelo para simular usuarios concurrentes.
  - Sistema Operativo: Se utilizará el sistema operativo Debian Linux 12 Bookworm en las instancias EC2 para ejecutar las pruebas.
  - Configuración de Red: Las instancias EC2 proporcionan una conexión de red hasta 10Gbps para hacer frente al ancho de banda necesario para enviar los videos.
- **Entorno GCP**
  - Google Cloud Monitoring: Utilizamos Google Cloud Monitoring para supervisar y gestionar nuestro entorno en la nube.
  - Compute Engine: Se ha implementado Compute Engine con una instancia e2-highcpu-2 que utiliza Debian 12. Esta máquina virtual cuenta con etiquetas para permitir el tráfico HTTP desde el Load Balancer y Health Check y actúa como receptor de información relacionada con la base de datos.
  - Instancia worker: Se ha configurado una instancia worker que ejecuta dos contenedores: uno con Redis y otro con Celery. Esta configuración optimiza la gestión de tareas y procesos.
  - Cloud Storage: Para el almacenamiento de videos, hemos integrado Cloud Storage, aprovechando su robusta capacidad de almacenamiento en la nube.
  - Compute Engine: Se ha implementado otra instancia Compute Engine llamada "monitoring-worker", utilizando una instancia e2-highcpu-2, para ejecutar las pruebas de estrés del escenario 2.
  - Cloud SQL Postgres: Para implementar una base de datos Postgres con 1 vCPU y 4 GB de memoria, garantizando un rendimiento eficiente y escalable.
  - Load Balancer: Integramos un Regional external Application Load Balancer para distribuir equitativamente la carga de tráfico entre las instancias, optimizando así el desempeño y la disponibilidad del sistema.
  - Managed Instance Group e Instance Template: Se ha configurado un Managed Instance Group y un Instance Template para gestionar y escalar fácilmente las

instancias, proporcionando flexibilidad y alta disponibilidad en nuestro entorno en la nube. Las instancias del Managed Instance Group se basan en el Instance Template.

- Autoscaling: Se activa autoscaling para el Managed Instance group, así que crean/eliminan instancias de acuerdo con la carga. Cuando se supera el objetivo del 55% de uso promedio de memoria, se crea una nueva instancia hasta un máximo de 3 instancias. Cuando se cae por debajo de nuevo, se elimina una instancia hasta el mínimo de una instancia.
- Pub/Sub: Es un servicio de mensajería y publicación/suscripción en GCP. Puede estar involucrado en la comunicación entre componentes del sistema.

- Aplicación Flask

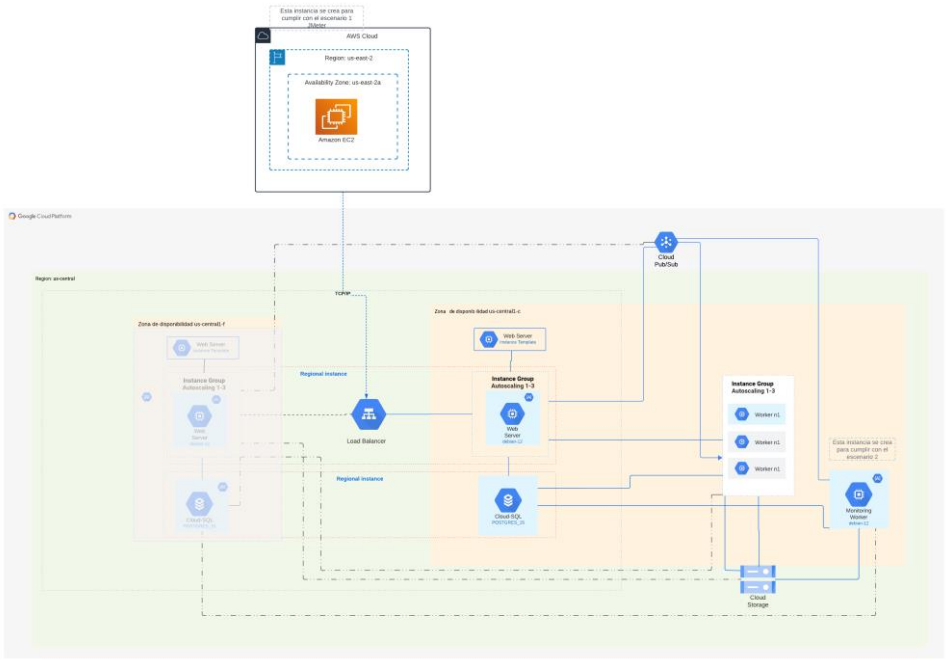

Imagen 1: Arquitectura

Escenarios

Se han identificado dos escenarios claves.

- **Escenario 1 – API REST de agregar una tarea**

  Descripción de funcionalidad

  Este servicio permite a los usuarios cargar un archivo multimedia que desean convertir a otro formato. Deben proporcionar el archivo a convertir y especificar el formato de destino. El archivo se almacena en la plataforma y se inicia el proceso de conversión. Se requiere un token de autenticación para utilizar este servicio.

  Para la conversión se agregan las tareas a una cola de mensajería, de donde luego serán procesados.

  Objetivo

Análisis de desempeño de procesar una solicitud del API REST de agregar una tarea (POST /api/tasks), almacenar el vídeo, actualizar la base de datos y agregarlo a la cola de tareas, así que quede listo para ser procesado por el componente worker.

Métricas

- **Throughput**: cantidad de peticiones procesadas por minuto.
- **Tiempo de respuesta (P95)**: percentil 95% de tiempo máximo que tarda la aplicación en procesar una petición
- **Tiempo de respuesta (P99)**: percentil 99% de tiempo máximo que tarda la aplicación en procesar una petición
- **Utilización de recursos**: monitoreo de la CPU, memoria y uso de red durante las pruebas.

Herramientas para la prueba

Se utilizará JMeter como la herramienta principal para realizar las pruebas de capacidad. Se realizarán 1000 peticiones con grupos concurrentes de 100 hasta llegar al tope.

Criterios de aceptación

- **Tiempo de Respuesta:**

  El tiempo de respuesta de la aplicación en todos los escenarios de prueba no debe superar los 0.5 segundos en el 95% de las transacciones, por petición.

  El tiempo de respuesta en ningún escenario de prueba debe superar los 4 segundos en el 99% de las transacciones.

- **Throughput:**

  La aplicación debe ser capaz de procesar 100 peticiones por minuto.

- **Utilización de recursos:**

  Durante las pruebas con 100 peticiones concurrentes, la CPU del servidor alcanza un pico de 80% y la memoria se mantendrá < 80% de uso.

Datos de prueba

- Se utilizarán archivos multimedia de ejemplo que representen escenarios reales de uso con un tamaño hasta 10MB.

Iteraciones

- Se realizarán 10 iteraciones

Riesgos

- Sobrecargar el entorno de producción, lo que podría resultar en una degradación del servicio o incluso una interrupción.
- Aprendizaje de la herramienta

- **Escenario 2 - Conversión de videos**

Descripción de funcionalidad

La funcionalidad implica la conversión de videos en formato MP4 a WebM, esta conversión es necesaria para adaptar videos a diferentes requerimientos de visualización.

Objetivo

Análisis de desempeño de convertir videos, actualizar el estado en la base de datos y guardar el video convertido, así que quede listo para ser recogido por el usuario.

Métricas

- **Throughput**: Cantidad de peticiones por minuto en la conversión de videos.
- **Tiempo de respuesta promedio:** Tiempo promedio que tarda la aplicación en procesar las tareas de conversión**.**
- **Tiempo de respuesta (P95)**: Percentil 95% de tiempo máximo que tarda la aplicación en procesar una tarea

Herramientas para la prueba

- **Conversor de video**
- **Script Python**: Se creará un script en Python que envíe datos a la cola de mensajería, simulando múltiples solicitudes concurrentes al sistema de conversión de videos.

Criterios de aceptación

- **Tiempo de Respuesta:**
  - El tiempo de respuesta promedio por solicitud de conversión no debe exceder los 40 segundos.
  - El tiempo de respuesta por petición no debe exceder los 120 segundos en más del 5% de las solicitudes.
- **Throughput:**
  - El sistema debe ser capaz de manejar un mínimo de 100 solicitudes por minuto para la conversión de videos MP4 a WebM.
- **Utilización de recursos:**
  - Durante las pruebas de conversión de video, el uso promedio de la CPU del servidor no debe exceder el 80% de su capacidad.
  - El uso promedio de la memoria RAM del servidor no debe exceder el 80% de su capacidad durante la conversión de videos.

Datos de prueba

- Se utilizarán archivos multimedia de ejemplo que representen escenarios reales de uso con un tamaño hasta 12MB.

Iteraciones

- Se realizarán 5 iteraciones

Riesgos

- Sobrecargar el entorno de producción, lo que podría resultar en una degradación del servicio o incluso una interrupción.
- Aprendizaje de la herramienta
- Este escenario se debe realizar mediante un script