Proyecto Final

Término: 2022-I

IMateria: Sistemas Distribuidos y Computación en la Nube

Número de integrantes: 1, 2 ó 3

Objetivos

En este proyecto los estudiantes:

- Aplicarán conceptos aprendidos en la materia a una aplicación distribuida real.
- Realizarán una evaluación de rendimiento a uno o más componentes de una aplicación distribuida.
- Tendrán la oportunidad de aprender a utilizar nuevos componentes de sistemas distribuidos (middlewares, librerías, bases de datos, servicios en la nube de AWS/Google/Azure, etc.).
- Demostrarán su capacidad de comunicación en un informe técnico en el que documentarán el diseño de su aplicación y los resultados de la evaluación de rendimiento.

Descripción

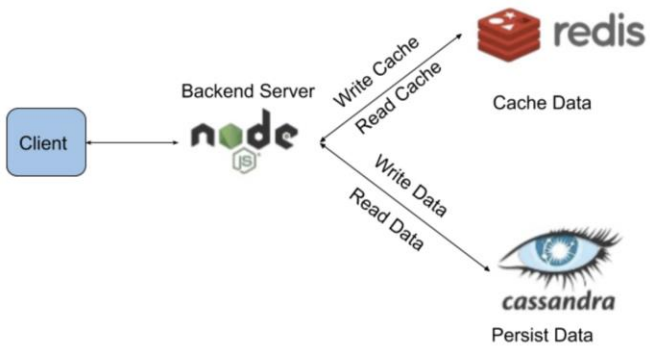
Para este proyecto usted deberá trabajar con un proyecto que **usted mismo** haya desarrollado este semestre o en semestres anteriores (en cualquier materia; la única restricción es que sea una aplicación distribuida; no importa si es una aplicación cliente/servidor básica). A dicho proyecto lo deberán mejorar aplicando conceptos aprendidos en la materia. Luego deberán realizar una evaluación de rendimiento (ej, usando herramientas de benchmarking o pruebas de stress) del nuevo diseño vs. el diseño original, de tal manera que podamos ver si el rendimiento ha mejorado. En caso de que la mejora que ustedes hayan realizado no sea de rendimiento sino una mejora de confiabilidad (reliability, tolerancia a fallos), entonces deberán realizar pruebas introduciendo fallos al sistema y deberán mostrar cómo el sistema se recupera de dichos fallos automáticamente.

Para asegurar que la mejora a su aplicación sea adecuada, proponemos una lista de mejoras posibles, de las cuáles su grupo debe elegir implementar una (o más si lo desean, con posibilidad de puntos extra):

- Rediseñar su backend utilizando una arquitectura de microservicios. Los microservicios deben correr en containers de Docker. Al menos uno de los microservicios debe estar replicado y deben usar Docker Swarm o Kubernetes para asegurarse que se levante automáticamente nuevas instancias de cualquier microservicio que falle.
- Si su proyecto anterior es una aplicación (web, móvil, cliente/servidor) implementada de manera monolítica, mejorar su rendimiento levantando múltiples réplicas de la aplicación y añadiendo un proxy / balanceador para que distribuya los requerimientos entre las múltiples réplicas (ej: NGINX o HAProxy). Cada réplica debe correr en un container de Docker. Esto requiere configuraciones simplemente. No debe requerir re-programación de su aplicación.
- Usar un orquestador de containers como Kubernetes o Docker Swarm para añadir tolerancia a fallos a algún componente de su sistema (frontend o backend); la manera de hacer esto es configurar el orquestador para que monitoree cuántas instancias están corriendo de cada contenedor y si de alguno hay cero instancias, entonces el orquestador automáticamente

levanta otra instancia. Esto requiere configuraciones solamente. No debe requerir re-programación de su aplicación.

- Añadir una caché (Redis) para acelerar los accesos a datos en la base de datos de su sistema. Por ejemplo, en lugar de que su aplicación lea directamente de la base de datos, su aplicación debe leer de la caché:



- Modificar su arquitectura para que use servicios en la nube, **administrados por el proveedor**. Ejemplo: cambiar la base de datos por Aurora, correr los componentes en un servicio de containers como ECS / EKS, etc. Al ser administrados por el proveedor, no es suficiente con subir su aplicación a la nube usando máquinas virtuales en EC2; este proyecto requeriría un rediseño y re-implementación total o parcial de su aplicación. Como proveedor en la nube se puede usar solamente AWS o Azure o GCP. Cabe notar que tanto Azure como AWS tienen un programa de créditos en la nube ($$) gratuitos para estudiantes. El programa de Azure es más generoso pero no estoy segura de qué servicios permite usar. El de AWS permite usar casi todos los servicios de AWS.
- Cambiar el API REST de una aplicación web o móvil por un API serverless usando API Gateway y AWS Lambda. Esta alternativa debe ser más escalable y más tolerante a fallos que el diseño original.
- Tema abierto: propuesto por el estudiante y aprobado por la Profesora. La aprobación debe realizarse máximo hasta tres semanas después del examen parcial. Enviar mensaje vía Teams con los detalles de la propuesta.

IMPORTANTE: No lo dejen la fase de pruebas para el último día porque no les alcanzaría el tiempo y perderían esos puntos. La evaluación es el componente más importante del proyecto, y como tal, deben darle la importancia que requiere.

Entregables

Cada grupo deberá subir un informe en formato PDF con las siguientes secciones:

1. [REQUERIDO] Título y URL al repositorio de Github con el código del proyecto. --> Esto no tiene puntaje, pero si no colocan el URL al repositorio, entonces el proyecto no será calificado.
2. [REQUERIDO] Listado de miembros del grupo **INCLUYENDO** una descripción de las actividades que realizó cada uno dentro del proyecto --> Esto no tiene puntaje, pero si no lo colocan, entonces el proyecto no será calificado.
3. [1 pt] Descripción de la aplicación (para qué sirve, para qué materia la implementaron).
4. [1 pt] Diseño original, incluyendo diagrama y descripción de los componentes.

5. [3 pts] Nuevo diseño, incluyendo diagrama, descripción de los cambios realizados y justificación/propósito de los mismos.
6. Evaluación de rendimiento **O** evaluación de tolerancia a fallos.
    a. [1 pt] Descripción del ambiente de pruebas, incluyendo información de las máquinas (físicas o virtuales) utilizadas, capacidad en RAM/CPU, sistemas operativos, etc.
    b. [3 pts] Diseño de pruebas: descripción de cómo realizaron las pruebas y cuál fue el propósito de las mismas, y qué métricas evaluaron. Por ejemplo, pueden evaluar latencia o throughput o tiempo que le toma al sistema recuperarse de una falla. Incluir detalles de las herramientas usadas para las pruebas (ej, para pruebas de stress: JMeter, bombardier, httperf, etc. Otras herramientas en: https://gist.github.com/denji/8333630). Las pruebas de stress sirven para evaluar escalabilidad y rendimiento. O, para introducir fallos, se puede usar herramientas como Chaos Monkey (https://github.com/Netflix/chaosmonkey) o scripts que desarrollen ustedes mismos (ej: un script en bash que usa ssh y kill para matar contenedores / procesos / componentes de su sistema).
    c. [7 pts] Resultados de los experimentos: Se evaluará que utilice gráficos adecuados. Por ejemplo, la latencia la debe mostrar como un CDF o un boxplot. El throughput lo puede mostrar como una serie de tiempo. Los ejes deben estar correctamente identificados. Las unidades de medida deben constar en los ejes. Hay muchos ejemplos en Internet de cómo graficar adecuadamente los resultados de pruebas de rendimiento. Para tolerancia a fallos, las series de tiempo son ideales: permiten mostrar el comportamiento antes del fallo, durante el fallo y luego cuando ya se recupera el fallo. Esta sección debe tener de 2 a 5 gráficos, dependiendo de lo que sea más adecuado para su proyecto. NO se aceptan screenshots de herramientas gráficas. Ustedes deben tomar las mediciones de su sistema y graficarlas usando R o Matplotlib o alguna herramienta equivalente.
    d. [3 pts] Análisis de los resultados. ¿Qué nos muestran los resultados?, ¿Mejoró el rendimiento? ¿Mucho/poco/nada? ¿Por qué?, ¿Mejoró la tolerancia a fallos?
7. [4 pts] Retrospective: Descripción de lo que aprendieron en el proyecto, incluyendo una lista de los principales problemas que se les presentaron y cómo los resolvieron.
8. [2 pts] Mejoras futuras: En base a lo que han aprendido en la materia, ¿qué otras mejoras sería recomendable hacerle a su aplicación? Justifique su respuesta.
9. [REQUERIDO] Referencias bibliográficas. --> Esto no tiene puntaje, pero si no colocan esta sección, entonces el proyecto no será calificado.

IMPORTANTE: Cualquier texto que esté escrito en el reporte debe ser autoría de alguno de los miembros del grupo. NO pueden copiar/pegar texto de Internet. Pueden parafrasear material de Internet si lo consideran necesario, pero es importante que lo parafraseen y que incluyan la referencia a dónde sacaron el material. **Reportes con copia o plagio serán reportados a la comisión de disciplina.**