Trabajo Práctico Nº 2

Sistemas distribuidos, comunicación y sincronización

Fecha de Entrega: 22/06/2020

Requisitos a tener en cuenta:

- Cada alumno o grupo de alumnos debe desarrollar las actividades prácticas enunciadas.
- Además, se debe generar un informe como resultado del trabajo realizado. Este debe incluir:
    a) Respuesta a las consultas, evaluaciones de métricas y tiempos, gráficas, conclusiones y, cuando corresponda, propuestas de mejoras sobre los servicios solicitados
    b) Pasos a seguir detallados para poder correr los ejercicios a la hora de realizar la corrección.
- El código de la aplicación tiene que estar subido a un repositorio tipo GIT (Github, Bitbucket, Gitlab) accesible desde Internet para tomar los recursos.
- Además del código fuente de la aplicación, la misma debe estar "compilada" para ser ejecutada directamente sin tener que abrir ningún IDE y preparada para correrse directamente desde la terminal (.jar, .war o aplicación contenedorizada).

Recomendaciones:

* Gestionar y mantener un registro de actividades y operaciones (Logs) en memoria (volátil) y disco (no volátil). Este apartado será contemplado positivamente a la hora de la evaluación.

* Trabajar teniendo en cuenta aspectos de separación de ambientes para que la solución pueda adaptarse a diversos entornos y configuraciones de red (no localhost, no cableado, ambientes híbridos). Para ello, se pueden utilizar archivos de configuración, servicios de descubrimiento, repositorios externos, entre otros.

* Integrar herramientas y soluciones que permitan garantizar servicios de alta disponibilidad, tolerancia a fallas y escalabilidad. Se recomienda el uso de:

    * Sistemas de colas para manejo de tareas asíncronas. Por ejemplo RabbitMQ.

    * Sistemas de bases de datos para el manejo de información estadística: Por ejemplo MariaDB.

* Portabilidad de entornos de trabajo: Por ejemplo Docker.

* Infraestructura Escalable y TF: Por ejemplo Kubernetes.

* Arquitecturas Híbridas: Por ejemplo AWS y recursos On Premise.

* Automatizaciones para la gestión de Infraestructura: Por ejemplo Terraform.

* Automatización en build, test y deploy del código: Por ejemplo Jenkins.

* Sistemas de análisis de logs y monitoreo: Por ejemplo ELK, Grafana y Prometheus.

Práctica

1) Desarrolle una red P2P de carga, búsqueda y descarga de archivos siguiendo las siguientes pautas:

- Existen dos tipos de nodos, Maestros y Extremos. Los primeros, son servidores centralizados replicados (al menos 2 nodos) que disponen del listado actualizado de los nodos extremos y se encargan de gestionar la E/S de los peers. Los segundos cumplen dos funciones en el sistema: realizan consultas (como clientes) y atienden solicitudes (como servidores).
- Funcionamiento:
  -- Cada extremo dispone de un parámetro definido en un archivo de inicialización con las direcciones IP de los nodos Maestros. Al iniciarse se contacta con un maestro el cual funciona como punto de acceso al sistema e informa cuáles son los archivos que dispone para compartir. Luego, está atento a trabajar en dos modos (cliente y servidor)

  -- Como cliente, deriva consultas al nodo maestro y una vez obtenida la respuesta, seleccionará el/los recursos que desee descargar y se contactará con el par correspondiente para descargar el/los archivo/s.

  -- Como servidor, recibe la consulta, revisa si matchea la consulta con alguno de los recursos disponibles y devuelve los resultados al nodo que solicitó resultados.

  A partir de los conceptos vistos en la teoría, critique este modelo y presente mejoras en su propuesta (que deben ser implementadas).

2) Un banco tiene un proceso para realizar depósitos en cuentas, y otro para extracciones. Ambos procesos corren en simultáneo y aceptan varios clientes a la vez. El proceso que realiza un depósito tarda 40 mseg entre que consulta el saldo actual, y lo actualiza con el nuevo valor. El proceso que realiza una extracción tarda 80 mseg entre que consulta el saldo (y verifica que haya disponible) y lo actualiza con el nuevo valor.

a) Escribir los dos procesos y realizar pruebas con varios clientes en simultáneo, haciendo operaciones de extracción y depósito. Forzar, y mostrar cómo se logra, errores en el acceso al recurso compartido. El saldo de la cuenta puede ser simplemente un archivo de texto plano.

b) Escribir una segunda versión de los procesos, de forma tal que el acceso al recurso compartido esté sincronizado. Explicar y justificar qué partes se deciden modificar

3) Construya una red flexible y elástica de nodos (servicios) la cual se adapte (crece / decrece) dependiendo de la carga de trabajo de la misma. El esquema será el de un balanceador de carga y nodos detrás que atienden los pedidos. Para ello deberá implementar mínimamente:

- Carga de sistema. Creación dinámica de conexiones de clientes y pedidos de atención al servicio publicado.
- Protocolo de sensado para carga general del sistema. Elija un criterio para detectar esa carga y descríbalo. Ejemplo: cantidad de clientes en simultáneo que el servicio puede atender. Si se excede esa cantidad, el punto de entrada (balanceador de carga) crea dinámicamente un nuevo servicio.
- Definición de umbrales de estado {sin carga, normal, alerta, crítico}
- Creación, puesta en funcionamiento de los servicios nuevos, y remoción de ellos cuando no sean más necesarios. Para esto el balanceador puede contar con una lista de IPs donde los servicios están instalados y pueden correr. De forma tal que arrancando inicialmente con 2 servicios en 2 nodos distintos, el sistema escala dinámicamente en función de la carga del sistema, usando los nodos listados en ese archivo de configuración. Si fuera necesario, puede haber más de un servicio en un mismo nodo. El servicio debe ser multi thread.

4) El operador de Sobel es una máscara que, aplicada a una imagen, permite detectar (resaltar) bordes. Este operador es una operación matemática que, aplicada a cada pixel y teniendo en cuenta los pixeles que lo rodean, obtiene un nuevo valor (color)

para ese pixel. Aplicando la operación a cada pixel, se obtiene una nueva imagen que resalta los bordes.

a) Desarrollar un proceso centralizado que tome una imagen, aplique la máscara, y genere un nuevo archivo con el resultado.

b) Desarrolle este proceso de manera distribuida donde se debe partir la imagen en n pedazos, y asignar la tarea de aplicar la máscara a N procesos distribuidos. Después deberá juntar los resultados. Se sugiere implementar los procesos distribuidos usando RMI.
A partir de ambas implementaciones, comente los resultados de performance dependiendo de la cantidad de nodos y tamaño de imagen.

c) Mejore la aplicación del punto anterior para que, en caso de que un proceso distribuido (al que se le asignó parte de la imagen a procesar) se caiga y no responda, el proceso principal detecte esta situación y pida este cálculo a otro proceso.