Apuntes clase presencial Bases de datos 2

Fecha: 26/09/2023

Estudiante: Mario Lara Molina

Dudas del proyecto

Cómo se expone la interfaz desde un contenerdor?

> Para la interfaz en un contenedor se expone el puerto de la máquina y se visualiza desde el navegador, se sube la imagen a docker hub y luego desde la máquina se jala la imagen
> Es el mismo ejemplo de cómo entrar a flask

La GUI se puede hacer en flask?

> Si funciona bien se puede, pero no ha visto buenos ejemplos de eso

Todo puede correr en una sola máquina virtual?

> Todo puede correr en una o varias máquinas virtuales

Porqué se usa una Rest api?

> La idea de una REST API es que hable JSON porque todos los lenguajes hablan JSON

El api y la gui se deben conectar en una sola conexión de docker?

> No porque todo se expone de manera pública mediante un url

La UI y la API no se conectan?

> No, desde el UI se llama a la API para procesar mediante un end point.

Deficiencias que tiene una base de datos sql contra una no sql, no podemos confiar totalmente en eso por la diferencia de hardware que tienen cada una de ellas, para tener una comparación real se debe usar el mismo hardware

Cuando tenemos un cloud provider tiene una LAN, normalmente es un lugar privado, dentro de la LAN se genera un dmz, que es una zona desmilitarizada, que es donde viven las partes de la apliación que se expone al público, por lo que si se hace una GUI que se renderize desde el servidor nunca se debe pasar código que se ejecute en el browser, se pone el GUI en la DMZ y la API se coloca en una red privada, para tener más controles. El acceso interno siempre se comunica por una red privada entre GUI y API para seguridad.
Existe otra red más privada que es donde se ponen a correr las bases de datos.

Lo que vamos a hacer es que desde la computadora se entra el GUI y al API, por lo que ambos son públicos.

Materia de clase

En bases de datos SQL se confunden con los nombres entre los diferentes elementos, en bases de datos no sql un nodo es igual a:

- Instancia
- Contenedor: Es porque dentro de un contenedor en Docker
- Pod: Es porque corre dentro de Kubernetes
- Servidor: Se piensa que es un servidor físico

Un cluster está compuesto por nodos, y un nodo es una unidad que contiene memoria, disco y procesador.

Hardaware on premise

Hardware con premise, aplica a sql y no sql, es una hardware no virtualizado, por lo que tenemos nuestra memoria, cpu y disco conectado a la máquina, por lo que las operaciones de entrada y salida son muy rápidas.

Cuando tenemos hardware en cloud, tenemos memoria, cpu y un componente de red que conecta con el disco, por lo que es normal que este componente se vuelva un cuello de botella, por lo que el rendimiento cae mucho.

Con un harware en cloud se tienen 16000 operaciones de entrada y salida por segundo, mientras que con harware on premise se tienen más de medio millón de entrada y salida por segundo.

Cluster

Un cluster es una colección de nodos, servidores, instancias, pods en el caso de elastic search ,siempre usamos transport lever security por lo que todas las comunicaciones entre servidores siempre están codificadas.

Cuando se instala elastic se crea un keeper, que tiene una llave pública y privada, el certificado siempre es generado por alguien. Hay un grupo de servidores que reciben los keepers los cuales almacenan los certificate of autority (CA)

Cuando se hace el boostrap de la db hace que todos contra todos se empiezen a intercambiar los certificados, se intercambian la llave pública, cuando las reciben se verifican con el certificate of autority, y cuando se tienen 100% válidas se establece la comunicación, todo esto se conoce como TLS, lo son estándares internacionales que establecen como manejar distintos tipos de datos.

Cuando manejamos una base de datos y le metemos niveles de encriptación le metemos overhead a la base de datos.

Si manejamos información con la que se puede identificar a una persona (PII) se deben tener ciertos

controles con infraestructura, Unidad de proteción de datos de la europea, se debe pensar como un repositorio de datos que debe mantener cierta seguridad.

Elasticsearch pensó que una DB tradicional, tiene muchos problemas, pero son muy buenas, con bases de datos no sql se necesita un driver para comunicarse con esa base de datos, por que es un canal de comunicación que no es estandar, el driver recibe un statement y devuelve un result set.

Cuando se dió la explosión del uso de JSON en web se pasó de 3 capas a una aplicación multicapas, por lo que tenemos un backend que se encarga de hablar con la db mediante el driver, un api o bussines logic, tenemos un UI que se comunica con el api.

Los patrones de diseño son modelos que sirven como guía para la solución de problemas muy comúnes en el desarrollo de software, en la actualidad se utilizan patrones un tanto distintos, por ejemplo se usa un facade en medio del UI y el API, este facade es un patrón de diseño muy usado.

Desacoplar da ventaja, cuando se tiene un api y se desacoplan se puede intercambiar el api y el UI nunca se ve afectado.

Broker con evolución en microservicios

En elasticsearch la aplicación habla directamente con un rest api con seguridad que expone un endpoint http para intercambiar información.
La comunicación desde un cliente se da mediante un rest api
Siempre las bases de datos sql tiene el storage y el procesamiento de consultas juntos, por lo que ocurren bloqueos por entrada y salida al mismo tiempo.

Las bases de datos no sql inventaron los tipos de nodos, en elastic hay:

- Ingest node: Solo es de procesamiento.
- Data note: necesitan alta memoria, alto I/O, alta memoria, alto cpu, por las agregaciones, las cuales resumen los datos que consulto desde los charts en elastic.
- Master node: Siempre deben ser 3 o número impar

Por diseño en elastic cualquier nodo puede recibir request de usuarios, pero el ingest no contiene datos, por lo que implica que si recibe un query el nodo no sabe como hacerlo, entonces pide el request, abre un hilo y le manda los datos al nodo data, espera que se procesa y luego se los devuelve al usuario.

Un nodo master no sabe insertar ni guardar datos, el se encarga de la sincronización del estado de elastic, maneja replicas y sus estados, sabe dónde está todo pero no tiene información. Por lo que cuando le pedimos algo va y habla con los demás nodos.

El nodo data almacena todos los datos, no es potente para hacer procesamiento de datos, el ingest es potente para hacer procesamiento de datos para correr ingestion pipelines.
Son operaciones que realizamos sobre los datos que ingresan en la base de datos.

Un error común es poner un load balancer y ponen todos los nodos de todos los tipos detrás del load balancer y le dan el load balancer a la app.
Al desacoplar roles se tiene garantía de que no afecta la manera en la que se distribuyen los datos a los usuarios o la manera en la que se ingresan los datos en la base de datos.
Si revisamos elasticsearch queques, tiene un concepto de colas, entonces se define un threat pull por lo

que se define un máximo de conexiónes y un cube.

También existen otros tipos de nodos que no son tan comúnes, por ejemplo:

- nodos transform: convierte índices en resumenes,se usa cuando se tiene un tiempo específico de corte

- Nodos machine learning: hardware especiaal que realiza transformaciones de machine learning

La consistencia eventual consiste en que los datos dentro de distintas bases de datos son consistentes enventualmente, por lo que no se puede determinar el tiempo en el que los datos van a ser guardados en otras réplicas, lo que puede generar problemas al consultar datos obsoletos. Esto es un buen indicador de que cuando se quieren sincronizar tareas no es buena idea usar una base de datos con consistencia eventual.

Un OCR es un receptor óptico de caracteres, el cual funciona con ingresando una imagen y el OCR reconoce los caracteres dentro de la imagen,por lo que si se mete una imagen a un nodo machine learning se puede verificar la cantidad de veces que aparece un texto en específico dentro de la base de datos.
Para poder hacer ese tipo de trabajo se debe usar un hardware especial, por lo que son transformaciones especializadas en inteligencia artificial.

Data tiers

Se dividen:

Content

- Se busca que los datos puedan quedarse en memoria siempre.
- Al tener el índice en memoria principal garantiza velocidad al no paginar hacia disco.
- Se necesita tener acceso constante, alta retención, potencia para realizar consultas, poder de procesamiento / IO, procesos complejos, búsquedas y agregaciones, índices bajos.

Hot data tier

- Se utiliza para timeseries, la cual es información que son strings de eventos que vienen en el tiempo.
- Siempre se habla de que se mantienen la cantidad de eventos más reciente, por lo que el dataset no siempre está en hot.
- El trade of en una base de datos es optimizar la base para el use case que tiene la mayor cantidad de consultas, y le damos el peso de esperar al usuario que realiza consultas pocas comunes.
- Se necesitan los más buscados, lecturas y escrituras rápidas, generalmente utilizan discos de estado sólido.

Warm data tier

- Información no tan frecuente como los hots
- Datos de semanas recientes
- Actualizaciones permitidas pero no son muy frecuentes

- No es tan rápido como los hot
- Una o más réplicas shards

Cold data tier

- Información que no es tan buscada
- Costos bajos de almacenamiento
- Fully mounted indices/ searchable snapshots

Frozen data tier

- Partially mounted snapshots
  no son usados / no longer required