Apuntes clase virtual Bases de datos 2

Fecha: 04/08/2023

Estudiante: Mario Lara Molina

Datos

Se convierte esa información en un asset mediante herramientas de

La información se convierte en un asset mediante herramientas de procesamiento de datos y bases de datos no sql

Un dato se convierte en información y toma relevancia cuando nos puede dar cierta influencia en toma de decisiones dentro de una empresa.

Tres estados de datos:

- at rest: está guardada en un disco duro, o en algún dispositivo de almacenamiento así como un disco duro, llave maya o en la nube
- in use: Los datos están siendo utilizados por una computadora, por ejemplo cuando se lee un archivo y se cargan los datos en memoria principal.
- in transit: Es un dato que se está moviendo a través de una red de computadoras.

Hablando de seguridad los datos at rest son los más seguros, y los in transit los más vulnerables.

Arquitectura

Siempre tenemos tres características importantes:

- Disco
- Memoria principal
- CPU

Cuando tenemos la memoria principal, siempre tenemos un "mal necesario" que se llama Sistema Operativo, el cual es un componente de software que gobierna totalmente las operaciones de la computadora y corre en superusuario.
Luego tenemos una sección de la memoria que se llama "sección de usuario" que es la que se utiliza para ejecutar una aplicación.

Por ejemplo, si tenemos una aplicación como "word" y hacemos click en la aplicación lo que sucede es que aparece el ciclo de fetch, el cual utiliza la memoria principal, la unidad de control y el procesador aritmético matemático, cuando se ejecuta una instrucción se genera lo que conocemos como interrumpción, cuando esto sucede el sistema operativo "despierta" y revisa la interrupción, por lo que ve que el usuario dió click en una aplicación y vé que no puede traer la aplicación completa a memoria principal.
Por lo que aparecen los ejecutables, que son archivos de código que se dividen en "páginas".

El sistema operativo utiliza el término de procesos, que son parecidos a los que se ven en el task manager, estos procesos están en una lista cargados en memoria.

El sistema operativo agarra la primera página de la aplicación y la carga en memoria, entonces tiene un puntero donde se encuentra esta primera página, luego de esto el sistema operativo vuelve al ciclo de fetch, imaginando que todo funciona como un round robin, lo cual significa que todos los procesos se colocan en una lista circular y la tenencia del cpu se va pasando de forma temporal. Aquí aparece un concepto llamado quantum, el cual es el tiempo que se ejecuta una app en un ciclo del cpu que normalmente es de 100ms.

Puede suceder que la aplicación tiene el cpu y termina sus procesos pero todavía no termina su tiempo del quantum, cuando esto sucede se produce un error llamado **page fault**, este error consiste en que el procesador llega al final de la página y pide la siguiente instrucción, pero cuando hace esto ve que la siguiente página no está cargada en memoria y se activa la interrupción del error. Por lo que se tiene una lista de bloqueos por entrada y salida, esto porque el sistema operativo tiene que esperar para leer la información en disco y cargarla en memoria principal, lo que ocasiona que se quite de la lista de procesos que están listos para ser ejecutados.

El sistema operativo como entidad superior de la computadora tiene la función de asegurarse de traer la siguiente página de memoria y cargarla en memoria principal, pero el cpu es más rápido que el sistema operativo, por lo que si se espera a que el cpu traiga memoria desde disco el proceso se demora demasiado, por lo que aparece el concepto de DNA, el cual es hardware que tiene una parte importante dentro del rendimiento de la computadora, es un chip de Direct Memory Access, por lo que el sistema operativo instruye al DNA con una dirección de memoria, la cual es una página, y luego le indica en qué parte de la memoria principal colocarla. Este chip no se rige con el cpu, por lo que trabaja con el sistema operativo, por lo que el sistema operativo puede encargarse de la lista de procesos sin mucho esfuerzo.

Cuando el DNA termina de copiar datos a memoria principal ocurre una interrupción, entonces despierta el sistema operativo y agarra la página, la carga en la lista de procesos para ejecutar por el cpu y luego continua el ciclo de fetch, mientras tanto la página ya se encuentra cargada en memoria principal y el procesador puede accesar a ella más fácil.

Cuando se tiene una base de datos o cualquier sistema computacional se piensa que se tienen recursos infinitos y que todos los programas se ejecutan en paralelo, pero esto no es así, sino que se reparten de manera secuencial con el tiempo compartido del cpu.

El cpu es demasiado rápido comparado con la memoria principal, por lo que cuando cuando se ejecuta un programa y se trae información desde memoria principal, aparece un nuevo concepto llamado **memoria caché**, la ual es una sección de almacenamiento muy rápido a disposición del cpu.
Existe un error de "cache miss", el cual invoca una interrupción que levanta el sistema operativo, selecciona la página de la app y la saca de la lista de procesos, ejecuta un algoritmo de selección y carga la siguiente página en memoria, por lo que mientras más grande la caché menos probabilidad existe de que ocurra este error, esto ayuda a mantener la velocidad y el rendimiento del equipo computacional.

Por lo que hasta el momento la arquitectura consiste de:

- Disco
- Memoria

- Cpu con caché
- Sistema operativo

El problema de invocar al sistema operativo mediante una interrupción es que este es un proceso complejo, por lo que este vive en disco, memoria principal, y el procesador, el cual tiene registros limitados de elementos que permiten ejecutar instrucciones con la ALU, dentro de los registros existe uno que se llama Program counter, el cual para apuntar a la siguiente página dentro del ciclo de fetch.

Existe algo llamado **context switch** el cual es el proceso de devolver el valor de los registros a un estado anterior, este ocurre cada vez que expira un quantum, el problema de utilizar los registros incorrectamente es que se llena la memoria principal y existen más posiblidades de que ocurra un page fault. Una vez dicho todo esto podemos pensar porqué esto es importante para una base de datos, sin embargo, una base de datos se comporta como una aplicación dentro de la computadora, por lo que se puede ver su work load que es el tipo de trabajo que realiza.

Existe un concepto llamado **CPU bounded** que significa que la base de datos realiza muchas operaciones aritméticas, lo que se refleja en las consultas de usuarios que contienen muchos "where" y agregaciones, esto permite destacar que el computador utiliza demasiado el procesador, esto ocurre cuando los usuarios cambian o cuando la base de datos llegó a crecer demasiado o que ya no funciona.

Dentro de las bases de datos existe el concepto del tamaño de bloque, el cual se mapea a un archivo de datos, lo que significa que si tenemos una base de datos de 10gb estructurados, entonces sql procesa bloques de 1mb cada uno, por lo que se dice que el tamaño de bloque es de 1mb, esto implica que si un usuario solicita datos de la base de datos entonces sql busca el bloque que se requiere. Una recomendación es que el tamaño de bloque de la base de datos y el de página del sistema operativo sean iguales, esto para reducir las interrupciones de lectura y aprovechar más el quantum.

El concepto de localidad tiene un alto impacto en el rendimiento y en el comportamiento de los usuarios, la localidad se define como la probabilidad de que el usuario pida datos cercanos de los solicitados en una primera instancia, normalmente los motores de bases de datos definen el tamaño de bloque pensando en la localidad, esto para tratar de predecir cómo se comporta el usuario y tener los datos en memoria principal antes de que se soliciten para reducir la intervención del sistema operativo.

También existe un concepto de Threats o procesos, el cual se utiliza cuando se tienen múltiples usuarios utilizando una misma base de datos, un proceso dentro de una computadora tiene un hilo principal por el cual el server de cliente reliza la consulta al server de la base de datos y ejecuta la consulta. El modelo que se utiliza permite que solo una aplicación pueda utilizar un puerto a la vez, por lo que una aplicación de cliente no tiene acceso exclusivo a un puerto, entonces cuando la aplicación solicita una conexión se llama al sistema operativo y hace un thread con un puerto aleatorio y atiende la consulta. Al momento en que esto sucede SQL solicita a la app que se conecte con el nuevo puerto, por lo que cuando llega otra app el sistema operativo genera otro thread y repite el proceso.

Cada conexión consume un pedacito de memoria, por lo que en algún momento puede llenarse la memoria y generar el error de **context switch**, esto pasa a nivel de bases de datos y ataques DNS, por lo que mientras más threads mayos concurrencia y mayor probabilidad de agotar los recursos disponibles, por lo que el servidor de base de datos debe tener un límite de usuarios.

Existen momentos donde se puede tener memoria principal libre y que la base de datos no funcione, esto introduce el concepto de **file descriptor**, este es un parámetro configurado en el motor de base de datos, consiste de un strnig de datos dentro de la computadora que tiene un flujo de datos continuo,

dentro del sistema operativo cada file descriptor tiene un identificador y un número limitado de datos. Por lo que cada vez que se genera una conexión crea un file descriptor y un thread, entonces se puede llegar a consumir toda esta parte en memoria y bloquear la base de datos.

El sistema operativo genera un file system sobre el disco, que es una estructura de datos que genera archivos, a nivel de memoria se guardan las estructuras de datos de procesos requeridas para ejecutar los programas, como los file descriptors, por lo que el sistema operativo no va a utilizar más espacio del necesario para manejar una palabra de procesador.