UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

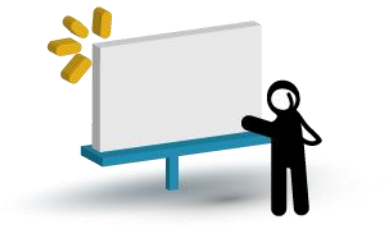
Centro de e-Learning

Diplomatura en Bases de Datos

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo 3: Herramientas de cara al futuro

Unidad 2: Bases de datos en memoria



Presentación:

En esta Unidad descubrimos qué es una base de datos en memoria. Discutimos, además, sus limitaciones, así como sus ventajas, para generar criterio sobre el tipo de situaciones en los que conviene aplicar esta herramienta. Finalmente, repasamos las distintas herramientas que ofrece el mercado.

Objetivos:

Que los participantes:

- Conozcan entiendan qué es una base de datos en memoria.
- Identifiquen en qué problemas conviene aplicar esta tecnología.
- Aprendan a seleccionar entre las distintas herramientas presentes en el mercado.
- Se familiaricen con los usos básicos de una base de datos en memoria.

Bloques temáticos:

1. Descripción del concepto de base de datos en memoria.
2. Ejemplos de bases de datos en memoria.

Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

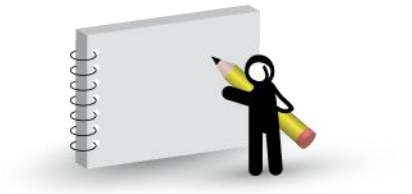
- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

**** El MEC es el modelo de E-learning colaborativo de nuestro Centro.***



Tomen nota:

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares.

Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación.

Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga.

Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas.

Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia.

Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

1. Descripción del concepto de base de datos en memoria

Una base de datos en memoria (IMDB), también conocida como base de datos de memoria principal (MMDB) es una base de datos cuyos datos se almacenan en la memoria principal para mejorar los tiempos de respuesta de las consultas, optimizando, de esta forma, el trabajo involucrado en el procesamiento de consultas.

Los datos de origen se cargan en la memoria del sistema en un formato comprimido no relacional, formando un tipo de base de datos analítica, en donde el sistema resulta de solo lectura que almacena datos históricos sobre métricas para aplicaciones de inteligencia empresarial / análisis empresarial (BI / BA), normalmente como parte de un DW. Estos sistemas permiten a los usuarios ejecutar consultas e informes sobre la información contenida, que se actualiza regularmente para incorporar datos de transacciones recientes de los sistemas operativos de una organización.

Además de proporcionar tiempos de respuesta de consulta extremadamente rápidos, el análisis en memoria puede reducir o eliminar la necesidad de indexación de datos y el almacenamiento de datos pre-agregados en cubos OLAP o tablas agregadas. Esta capacidad reduce los costos de IT y permite una implementación más rápida de las aplicaciones de BI / BA.

Tres desarrollos en los últimos años han hecho que el análisis en la memoria sea cada vez más factible: computación de 64 bits, servidores de múltiples núcleos y menores precios de RAM.

Por qué surge la necesidad de una base de datos en memoria?

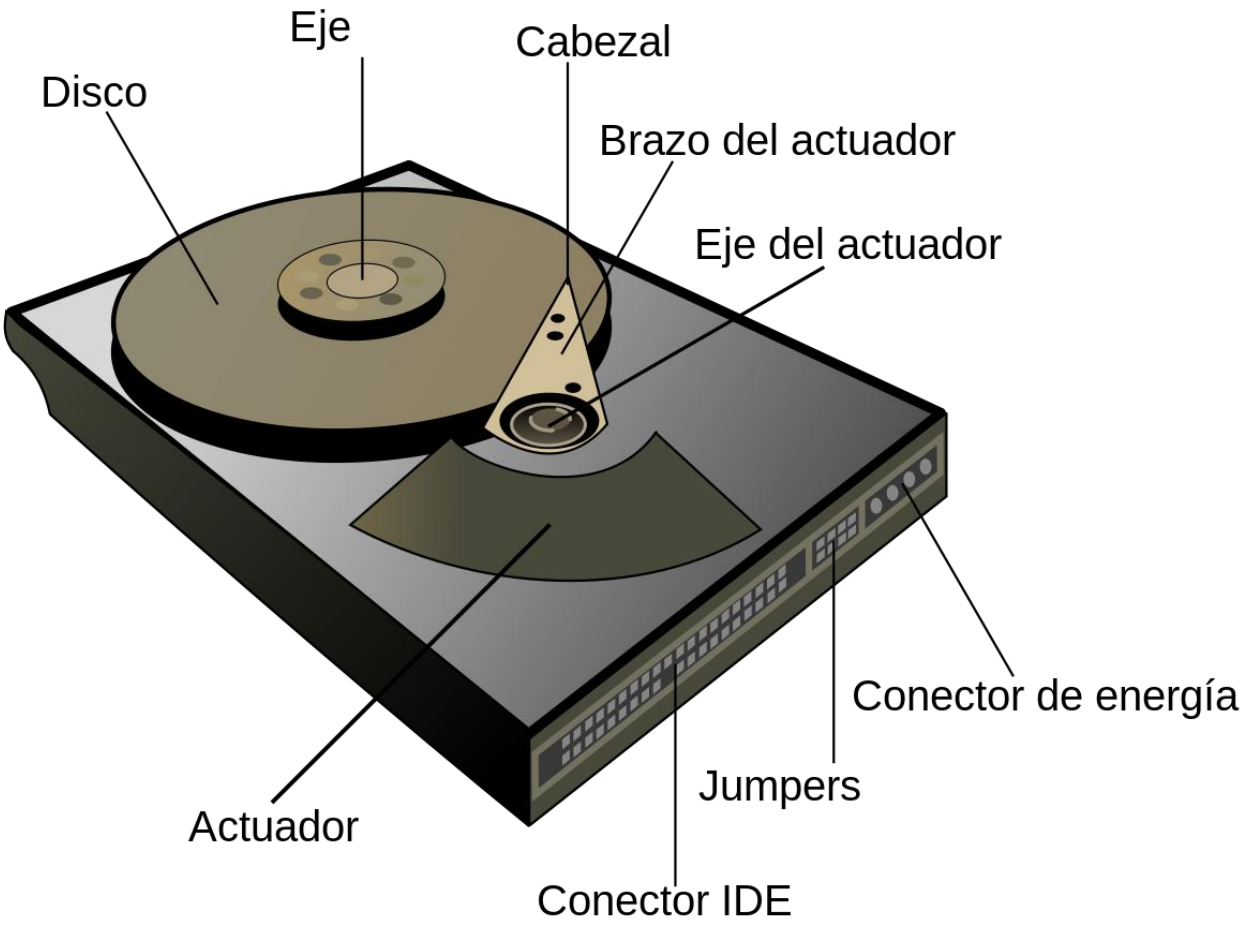
Desde los primeros sistemas de bases de datos, los profesionales de estos sistemas se han esforzado por evitar el proceso de IO en un disco. Los discos magnéticos han estado presente dentro de la informática digital desde la década de 1950.

Su arquitectura ha cambiado muy poco: uno o más platos contienen cargas magnéticas que representan bits de información. Estas cargas magnéticas son leídas y escritas por un

brazo actuador, que se mueve a través del disco a una posición específica en el radio del plato y luego espera a que el plato gire a la ubicación apropiada.

Este tipo de acceso a los discos magnético siempre ha sido de muchos órdenes de magnitud más lento que a la memoria o la CPU, y la situación solo ha empeorado ya que la ley de Moore aceleró el rendimiento de la CPU y las memorias mientras que dejó atrás el rendimiento del disco mecánico.

Este crecimiento, claramente, no se aplica a los aspectos mecánicos del rendimiento del disco. Por consiguiente, los discos magnéticos se han convertido en una carga cada vez mayor para el rendimiento de la base de datos. La aparición de la tecnología de disco de estado sólido (SSD) ha permitido un salto cualitativo en rendimiento del disco para las bases de datos.



A diferencia del disco magnético, los SSD no contienen partes móviles y proporcionan latencias de IO más bajas. Los SSD comerciales se implementan actualmente utilizando DDR RAM, que de hecho es una batería con respaldo de dispositivo RAM o flash NAND. El flash NAND es un medio de almacenamiento intrínsecamente no volátil y casi domina completamente el mercado actual del SSD.

Esta tecnología ha pasado de ser un costoso lujo a una tecnología convencional que tiene un lugar en casi todos los sistemas de bases de datos donde el rendimiento es crítico.

Así, los SSD tienen algunas características de rendimiento únicas y algunos sistemas de bases de datos se han desarrollado para explotar estas capacidades.

Por ejemplo, el rendimiento de la SSD flash es en órdenes de magnitud superior a los dispositivos de disco magnético, especialmente para operaciones de lectura. Una lectura aleatoria de un disco de estado sólido de gama alta puede completarse en tan solo 25 microsegundos, mientras que una lectura de un disco magnético puede tomar hasta 4000 microsegundos (4 milisegundos o 4/1000 de segundo), más de 150 veces más lento.

Mientras que las SSD son ciertamente más rápidas que los discos magnéticos, la mejora de velocidad no es proporcional para todas las cargas de trabajo. En particular, cuesta más, es decir, lleva más tiempo, modificar la información en un SSD que leer de él. Las SSD almacenan bits de información en las celdas.

Un SSD de un solo nivel de celdas (SLC) contiene un bit de información por ceda, mientras que una SSD de múltiples niveles (MLC) contiene más de un bit, por lo general solo dos, pero a veces tres en cada celda Las celdas se organizan en páginas de aproximadamente 4K y las páginas se organizan en bloques que normalmente contiene 256 páginas.

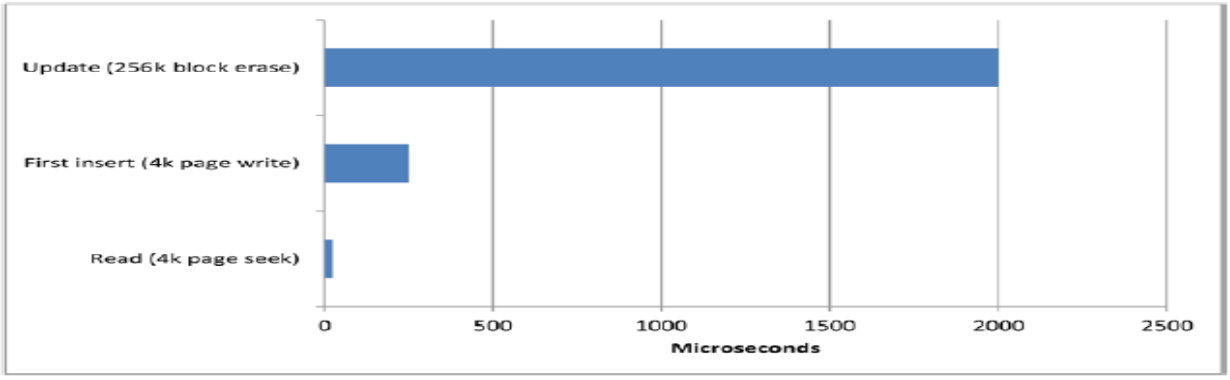
Las operaciones de lectura y las operaciones de escritura iniciales solo requieren un IO de una sola página. Sin embargo, cambiando el contenido de una página requiere borrar y sobrescribir un bloque completo.

Incluso la escritura inicial puede ser significativamente más lento que una lectura, pero la operación de borrado de bloques es particularmente lenta, alrededor de dos milisegundos. De ahí surge la necesidad de soluciones que no utilicen dispositivos de disco.

Mientras que las SSD permiten que el proceso de IO se acelere, la capacidad creciente y la economía de la memoria del servidor a veces permite evitar el proceso de IO por completo.

Muchas bases de datos más pequeñas ahora pueden caber completamente dentro de la capacidad de memoria de un solo servidor, y ciertamente dentro de la capacidad de memoria de un clúster. Para estas bases de datos, una solución en memoria puede ser aún más atractiva que una arquitectura SSD.

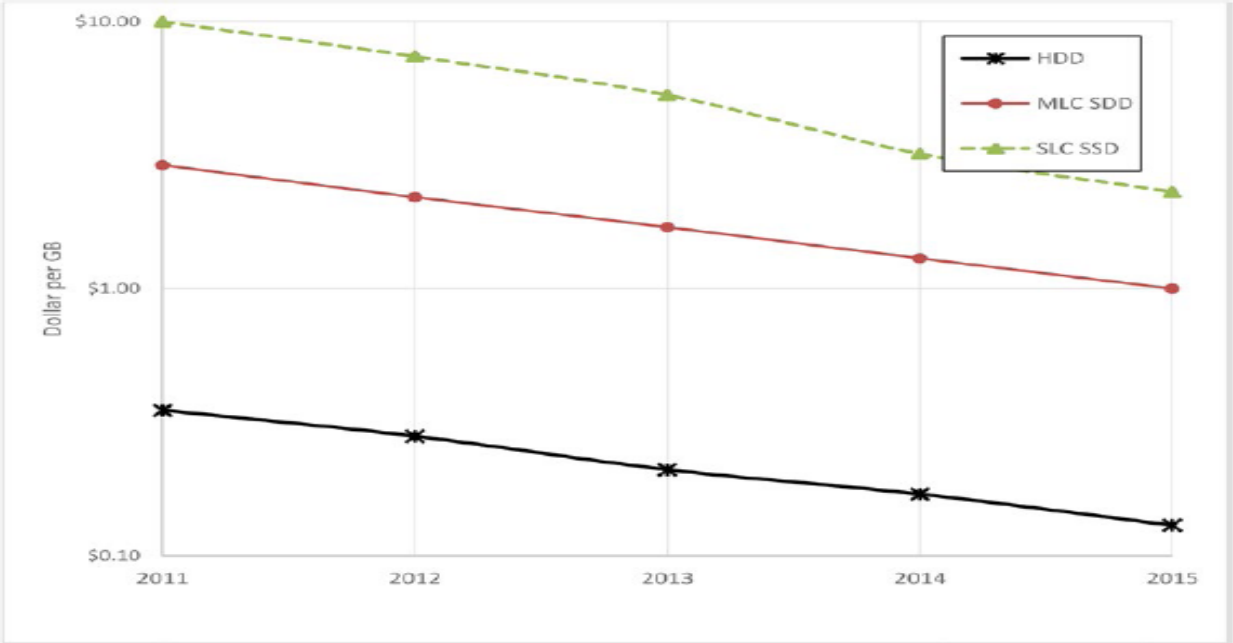
La Figura siguiente muestra los tiempos aproximados para una búsqueda de página, escritura de página y borrado de bloque en un SSD.



Guy Harrison, Next Generation Databases, 2016

Los fabricantes de SSD comerciales realizan grandes esfuerzos para evitar las fallas de rendimiento de la operación de borrado y se plantean soluciones para evitar la perdidad de confiabilidad por el desgaste que ocurre en un SSD cuando se modifica una celda. Para eso, se utilizan algoritmos sofisticados para garantizar que las operaciones de borrado se minimicen y que las escrituras sean distribuidas uniformemente en el dispositivo. Sin embargo, estos algoritmos amplifican la escritura incorporando varios procesos IO físicos en el SSD a medida que se mueven y se regeneran los datos. Esto provoca casi el mismo efecto que con los discos magnéticos: el flash SSD es significativamente más lento cuando se realizan escrituras que al realizar lecturas.

Por eso, la tecnología magnética del disco proporciona un medio más económico por unidad de almacenamiento, mientras que la tecnología flash proporciona un medio más económico para ofrecer altas tasas de I/O y bajas latencias. Y aunque el costo de un SSD está cayendo rápidamente, también lo está el costo de un disco magnético. Un examen de la tendencia del costo por gigabyte para SSD y HDD se muestra en la siguiente figura:



Guy Harrison, Next Generation Databases, 2016

Así el SSD resulta ser una una solución económica y eficaz para bases de datos chicas o para sistemas donde el rendimiento es crítico. Sin embargo no resulta completamente útil para bases de datos masivas, especialmente para datos a los que se accede con poca frecuencia.

Bases de datos habilitadas para SSD

Algunos sistemas de bases de datos de última generación han diseñado arquitecturas específicas para aprovechar la física del SSD. Por ejemplo, Aerospike es una base de datos NoSQL que intenta proporcionar una arquitectura de base de datos que puede aprovechar al máximo las características IO de un flash SSD.

Este administrador de base de datos no es de licencia pública, pero es muy rápido, altamente escalable y muy confiable para aplicaciones de big data y en tiempo real (*https://www.aerospike.com*). Se puede instalar en Amazon EC2, en diferentes distribuciones de Linux, OS X, Windows y en varios proveedores de servicios en la nube.

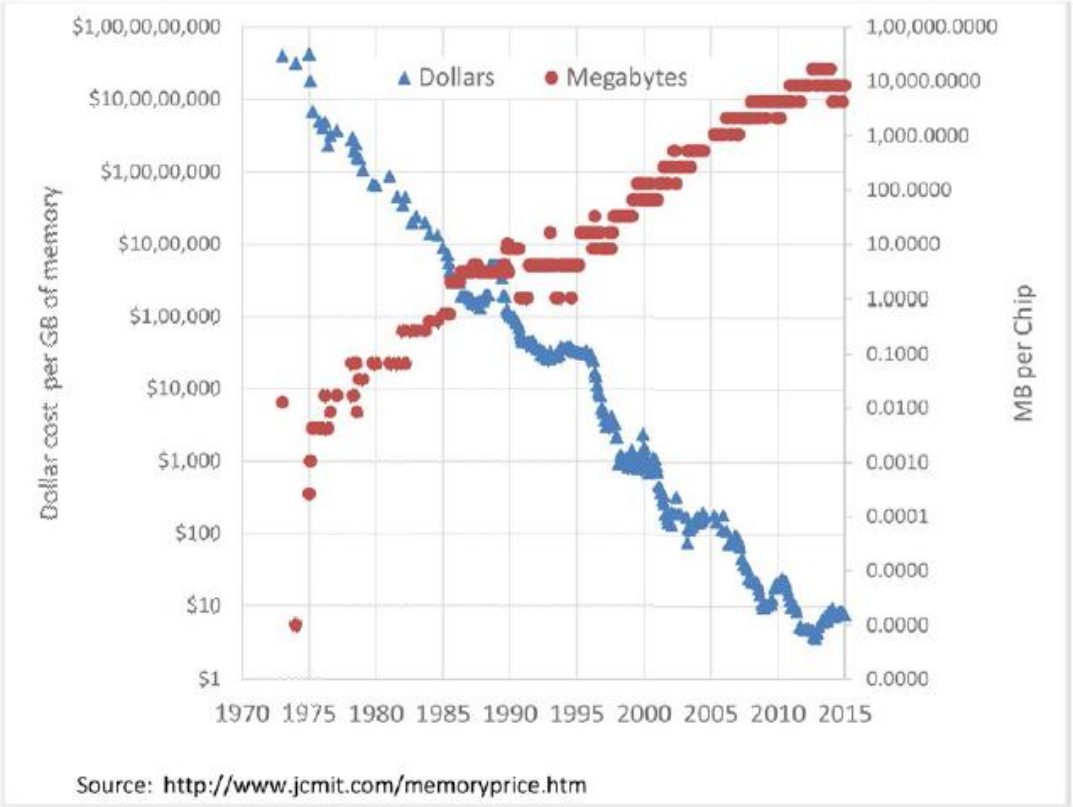
Aerospike implementa un sistema de archivos con estructura de registro en el que se actualizan implementado físicamente al anexar el nuevo valor al archivo y marcar los datos originales como no válidos. los el almacenamiento de los valores anteriores se recupera mediante un proceso en segundo plano en un momento posterior. Aerospike también implementa un enfoque inusual en el uso de la memoria principal. En lugar de usar main memoria como caché para evitar el disco físico IO, Aerospike utiliza la memoria principal para almacenar índices en los datos mientras manteniendo los datos siempre en flash.

Este enfoque representa un reconocimiento de que, en un sistema basado en flash, el El enfoque "evitar IO a toda costa" de las bases de datos tradicionales puede ser innecesario. De esta manera, Aerospike ha optimizado la base de datos para aprovechar el almacenamiento Flash al máximo, cuando estos dispositivos SSD se utilizan para el almacenamiento.

Bases de datos en memoria

El SSD puede haber tenido un impacto transformador en el rendimiento de las bases de datos, pero sus resultados no son aplicables a todo el espacio de bases de datos. Una tendencia más cambiante de paradigma ha sido la de aumentar la practicidad de almacenar bases de datos completas en la memoria principal.

Tanto el costo de la memoria como la cantidad de memoria que se puede almacenar en un servidor han variado notablemente desde los primeros días de la informática. La siguiente figura ilustra estas tendencias: tanto costo de memoria por unidad de almacenamiento y la cantidad de almacenamiento que puede caber en un solo chip de memoria han ido aumentando durante muchas décadas (observe la escala logarítmica: las líneas relativamente rectas indican tendencias exponenciales).



Guy Harrison, Next Generation Databases, 2016

El tamaño de la base de datos promedio, particularmente a la luz del fenómeno Big Data, ha estado creciendo exponencialmente también. Para muchos sistemas, el crecimiento de la base de datos continúa superando el crecimiento de la memoria. Pero muchas otras bases de datos de tamaño más modesto ahora pueden almacenarse cómodamente en la memoria de un único servidor.

Y muchas más bases de datos pueden residir dentro de la capacidad de memoria de un clúster. Las bases de datos relacionales tradicionales usan la memoria para almacenar en caché los datos almacenados en el disco, y generalmente muestran mejoras significativas en el rendimiento a medida que aumenta la cantidad de memoria. Pero hay algunas operaciones en bases de datos que simplemente deben escribir en un medio persistente. En la arquitectura de base de datos tradicional, las operaciones de tipo COMMIT requieren una escritura en un registro de transacciones en un medio persistente, y periódicamente la base de datos escribe bloques de "punto de control" en la memoria del disco.

Aprovechar al máximo un sistema de memoria grande requiere una arquitectura que es consciente de que la base de datos es completamente residente en memoria y que permite las ventajas de acceso de alta velocidad sin perder datos en caso de una falla de energía. Hay dos cambios en la arquitectura de base de datos tradicional que un sistema en memoria debe abordar:

- Arquitectura sin caché: las bases de datos tradicionales basadas en disco casi invariablemente almacenan en caché datos en la memoria principal para minimizar el IO en disco. Esto es inútil y contraproducente en un sistema en memoria: no tiene sentido almacenar en la memoria lo que ya está almacenado en la memoria.
- Modelo de persistencia alternativo: los datos en la memoria desaparecen cuando la energía es apagada, por lo que la base de datos debe aplicar algún mecanismo alternativo para garantizar que no ocurra pérdida de datos.

Así, las diferencias significativas que presentan las bases de datos relacional y las bases de datos en memoria son:

Bases de datos en disco

Todos los datos almacenados en disco, se necesitan IO de disco para mover datos a la memoria principal cuando es necesario.

Los datos siempre se conservan en el disco.

Las estructuras de datos tradicionales como B-Trees son diseñadas para almacenar tablas e índices de manera eficiente en el disco.

El tamaño de la base de datos es virtualmente ilimitado.

Admite un conjunto muy amplio de cargas de trabajo, es decir, OLTP, almacenamiento de datos, cargas de trabajo mixtas, etc.

Bases de datos en memoria

Todos los datos son almacenados en la memoria principal, no es necesario realizar IO de disco para consultar o actualizar datos.

Los datos son persistentes o volátiles dependiendo del producto de la base de datos en memoria.

Las estructuras de datos especializadas y las estructuras de índices suponen que los datos siempre están en la memoria principal.

Están optimizadas para cargas de trabajo especializadas; es decir, cargas de trabajo HLR / HSS específicas de la industria de comunicaciones.

El tamaño de la base de datos está limitado por la cantidad de memoria principal.

Las bases de datos en memoria generalmente usan una combinación de técnicas para garantizar que no pierdan datos. Éstas incluyen:

- Replicar datos a otros miembros de un clúster.
- Escribir imágenes completas de la base de datos (llamadas instantáneas o puntos de control) en los archivos del disco.
- Escribir registros de transacción/operación en un archivo de disco de solo anexar (llamado registro de transacciones o diario).

Necesidades y usos de las bases de datos en memoria

El uso principal de un sistema de administración de bases de datos en memoria es mejorar el rendimiento de las consultas y las aplicaciones que acceden a los datos. Pero además de esta necesidad de acelerar los procesos mediante el uso de la memoria, existe otras razones.

En primer lugar, la cantidad de memoria disponible en los servidores actuales continúa expandiéndose, y muchos servidores actuales tienen 32 terabytes (TB) de memoria o más.

Además, el costo de la memoria continúa disminuyendo, aunque sigue sin ser tan barata como el disco. Pero también, en segundo lugar, existe una necesidad de velocidad en las aplicaciones modernas que está contribuyendo al aumento del IMDBMS. Las organizaciones están creando y adoptando más aplicaciones en tiempo real y orientadas a la Web que pueden beneficiarse del rendimiento de alta gama que puede ofrecer la tecnología en memoria.

En este punto, la tecnología actual de procesamiento en memoria ayuda a que la persistencia ya no sea una barrera. Por su propia naturaleza, la memoria es una forma volátil de almacenamiento. Si el servidor pierde potencia, los datos en la memoria se perderán.

Sin embargo, las opciones modernas de IDBMS en memoria se han diseñado para que los datos en la memoria persistan incluso después de una interrupción. Por esta razón, este tipo de bases de datos se ha vuelto viable para los requisitos de procesamiento transaccional y analítico de la mayoría de las organizaciones.

Entre las ventajas, que se pueden mencionar, que ofrecen las IDBMS está, lógicamente, la mejora significativa del rendimiento cuando se compara con las alternativas basadas en disco. Dando ganancias de rendimiento de tres a cuatro, y algunas veces mucho más, que cuando se usa un sistema tradicional.

Si bien sas opciones de IMDBMS han sido tradicionalmente fuertes en el mercado de bases de datos integradas, donde es deseable tener una huella pequeña y una arquitectura no invasiva. Actualmente su mercado ofrece muchas opciones fuertes de calidad empresarial, es decir, mucho más que solo aplicaciones integradas.

Sin embargo, existe una desventaja principal que hace que no haya habido una migración masiva a este tipo de sistemas: el costo. Si bien los precios están disminución, la memoria aún cuesta más que el disco.

Sumado a este motivo, existe una la falta de experiencia en IMDBMS, y muchas implementaciones heredadas de DBMS junto con opciones no estándar que provocan que, para obtener la velocidad más alta, sea necesaria una interfaz diferente de SQL puro.

Además, el tamaño de la base de datos es un factor restrictivo. Y aunque los avances tecnológicos están eliminando esta restricción y las IMDBMS pueden manejar bases de datos muy grandes (escalan a más de un terabyte), muchos consideran que sólo pueden usarse en sistema de tamaño limitado.

Los casos de uso para IMDBMS son amplios y variados. Cualquier aplicación que pueda beneficiarse de un aumento en el rendimiento puede sacar provecho de usar un IMDBMS.

En particular, las aplicaciones con requisitos de administración de datos en tiempo real pueden beneficiarse, como aplicaciones para telecomunicaciones y redes, mercados de capitales, defensa e inteligencia, viajes y reservas, aplicaciones de centros de llamadas y juegos.

También son candidatas a poder usar las IMDBMS las aplicaciones con una necesidad inmediata de datos, como aplicaciones para inteligencia empresarial en tiempo real, detección de fraude, análisis en tiempo real y transmisión de datos.

Tipos de sistemas de base de datos en memoria

Una de las primeras formas de procesamiento de datos en memoria fue realizada por los programadores de COBOL que crearon tablas en memoria para almacenar datos a los que se podía acceder varias veces a medida que se ejecutaba un programa. Esto, por supuesto, no era un sistema de base de datos, pero era una forma temprana de acceso a datos en memoria adoptado para aumentar la velocidad de procesamiento.

Luego, surgieron técnicas para acceder a los datos desde la memoria en lugar del disco. En el nivel básico, los DBA deben configurar los niveles adecuados de memoria para almacenar en caché los datos en las agrupaciones de almacenamientos intermedios. El almacenamiento en caché permite que los datos permanezcan en la memoria para accesos posteriores.

También apareció otra forma de procesamiento de datos en memoria mediante el uso de los SSDs, mencionados al incicio de la unidad. En este caso, el almacenamiento de datos se basa en chips de memoria en lugar de girar el disco para almacenar datos de forma persistente, y que tiene una historia desde la década de 1950 en grandes computadoras centrales y supercomputadoras.

En la década de 1980, algunas bases de datos del sistema de gestión de la información se almacenaban en un tipo de SSD inicial. Pero hasta hace poco, la tecnología era prohibitiva en términos de costos para una adopción generalizada. Hoy en día, con la memoria más barata que nunca, una forma sencilla de lograr bases de datos en memoria puede ser simplemente almacenar archivos de bases de datos en SSD.

Pero las IMDBMS modernas están diseñadas específicamente para el procesamiento en memoria. Todos los datos se almacenan en la memoria (memoria de acceso aleatorio dinámica, o DRAM) en un servidor, y todas las operaciones se realizan en la memoria.

Todos los datos están en la memoria, y el IMDBMS no es simplemente un caché en memoria. Los datos pueden almacenarse en un formato comprimido para permitir un almacenamiento y acceso más eficientes.

Actualmente, existen bases de datos híbridas en memoria / disco. Un híbrido no solo se basa en los chips de memoria para almacenar los datos, sino también en las unidades de disco duro.

La ventaja de un IMDBMS híbrido es la flexibilidad, donde las bases de datos se pueden diseñar con un equilibrio de rendimiento, costo y persistencia. Muchas aplicaciones se pueden beneficiar del acceso rápido de algunos datos a la memoria, con otros datos a los que se accede menos frecuentemente almacenados en el disco.

El disco es aún más económico que la memoria, por lo que los intercambios posibles con una solución híbrida atraen a muchas organizaciones con requisitos mixtos y presupuestos más ajustados.

La mayoría de los principales proveedores de sistemas de administración de bases de datos relacionales (RDBMS) están agregando capacidades de base de datos en memoria para complementar el almacenamiento existente en disco.

Los sistemas de bases de datos NewSQL son otra rama de las tendencias actuales en memoria y NoSQL. El concepto de NewSQL es adoptar el impulso del mercado de NoSQL con arquitecturas de base de datos modernas, con configuraciones e implementaciones que admitan SQL, aprovechando así el conocimiento del enorme grupo de desarrolladores de SQL. No todos los productos NewSQL DBMS están en la memoria, pero muchos sí lo están.

Los productos IMDBMS pueden ser relacionales, NoSQL, NewSQL o cualquier otro tipo de DBMS. Se pueden usar para el procesamiento de transacciones operacionales o para aplicaciones analíticas de inteligencia empresarial. Por supuesto, cada producto en particular tendrá diferentes características y capacidades que le permitirán soportar el procesamiento operativo mejor que mejor que el procesamiento analítico (o viceversa).

2. Ejemplos de bases de datos en memoria

TimesTen

TimesTen es un sistema de base de datos en memoria temprana que aspira a soportar cargas de trabajo similares a las de un sistema relacional tradicional, pero con un mejor rendimiento. TimesTen fue fundada en 1995 y adquirida por Oracle en 2005. Oracle lo ofrece como una base de datos en memoria independiente o como una base de datos de almacenamiento en caché que complementa el RDBMS de Oracle tradicional basado en disco:

www.oracle.com/technetwork/database/database-technologies/timesten/overview/timesten-imdb-086887.html

TimesTen implementa un modelo relacional basado en SQL bastante familiar. Después de la compra por Oracle, implementó el estándar ANSI SQL, pero en los últimos años el esfuerzo ha sido hacer que la base de datos sea compatible con la base de datos Oracle central, hasta el punto de soportar el lenguaje de procedimientos almacenados PL / SQL de Oracle.

Actualmente, la última versión de TimesTen ofrece un rendimiento en tiempo real al cambiar las suposiciones sobre dónde se encuentran los datos en el tiempo de ejecución. Al administrar los datos en la memoria y optimizar las estructuras de datos y acceder a los algoritmos en consecuencia, las operaciones de la base de datos se ejecutan con la máxima eficiencia, logrando ganancias dramáticas en receptividad y rendimiento, incluso en comparación con un RDBMS en disco completamente en caché.

TimesTen puede integrarse dentro de las aplicaciones para mejorar aún más el rendimiento de las operaciones de la base de datos al eliminar la comunicación entre procesos y los gastos generales de red.

TimesTen se implementa comúnmente con aplicaciones multiusuario y de subprocesos múltiples mediante el bloqueo de nivel de fila y el aislamiento de lectura comprometida. Las aplicaciones acceden a las bases de datos TimesTen utilizando SQL estándar a través de las interfaces de programación JDBC, ODBC, ODP.NET, OCI (Oracle Call Interface), Pro * C / C ++ y Oracle PL / SQL.

Si bien el mejor tiempo de respuesta se logra con TimesTen ejecutándose en proceso con la aplicación (también conocido como "modo directo"), el acceso convencional de cliente / servidor se usa comúnmente cuando una base de datos es compartida por varias aplicaciones que se ejecutan en diferentes servidores.

Las bases de datos de TimesTen son persistentes y recuperables. La durabilidad se logra mediante una combinación de registro de transacciones y punto de referencia de la base de datos en el disco.

La alta disponibilidad se proporciona mediante la replicación TimesTen para replicar transacciones entre bases de datos en tiempo real. Debido a la naturaleza crítica de las aplicaciones TimesTen, la mayoría de las implementaciones empresariales utilizan la replicación TimesTen para la disponibilidad y recuperación de fallas.

Por ejemplo, las telecomunicaciones y los sistemas globales accesibles a través de la web, como la carga en línea, la administración de sesiones de suscriptores, las tiendas en línea de comercio electrónico, los sitios web de viajes y reservas no pueden tolerar el tiempo de inactividad de la aplicación / servicio; los servicios financieros y los sistemas de negociación de valores deben permanecer continuamente disponibles mientras los mercados financieros estén abiertos.

La replicación en TimesTen utiliza una tecnología basada en registros de transacciones optimizada para la memoria con un protocolo de red basado en flujos eficiente para un alto rendimiento, confiabilidad y robustez. Las características clave incluyen:

- Replicación asincrónica; proporciona el máximo rendimiento al desacoplar la ejecución de la transacción de aplicaciones de la captura, la propagación y aplicar los pasos de replicación.

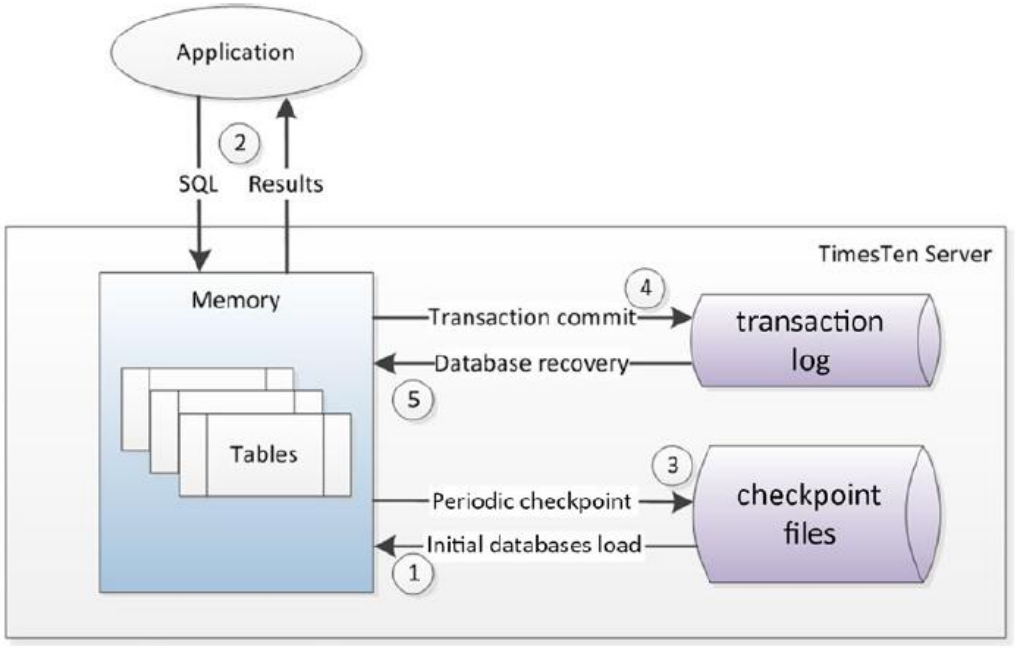
- Replicación sincrónica; proporciona un mayor nivel de protección de datos al mantener una coherencia total entre las bases de datos activas y en espera; una confirmación de la aplicación solo se devuelve cuando la transacción duplicada se ha recibido y confirmado en la base de datos en espera.
- Disponibilidad de lectura de la base de datos en espera; Se puede proporcionar capacidad de lectura adicional configurando suscriptores adicionales de solo lectura.
- Replicación paralela para aplicaciones que requieren un rendimiento de transacción muy alto.
- Detección automática de fallas de la base de datos en espera; logrado mediante una integración perfecta con Oracle Clusterware.
- Actualizaciones en línea; permitiendo que los servidores individuales se desconecten para las actualizaciones de software, mientras que otros servidores continúan ininterrumpidamente.

Así, en una base de datos TimesTen, todos los datos son residentes de memoria. La persistencia se logra escribiendo instantáneas periódicas de la memoria en el disco, así como escribir en un registro de transacciones basado en disco después de una confirmación de transacción. En la configuración predeterminada, todas las grabaciones de disco son asíncronas: para una operación en la base de datos no seería necesario esperar por una operación de disco IO.

Sin embargo, si falla la energía entre el proceso de transacción y la hora en que se escribe el registro de transacciones, los datos podrían perderse. Este comportamiento no es compatible con ACID porque la durabilidad de la transacción (la "D" en ACID) no está garantizada.

Sin embargo, el usuario puede elegir configurar escrituras sincrónicas en el registro de transacciones durante las operaciones de confirmación. En este caso, la base de datos se convierte en una base que cumple con ACID, pero algunas operaciones de base de datos esperarán en el IO del disco.

La arquitectura TimesTen es como la siguiente:



Guy Harrison, Next Generation Databases, 2016

Cuando se inicia la base de datos, se cargan todos los datos desde los puntos de control en la memoria principal (1).

La aplicación interactúa con TimesTen a través de solicitudes SQL que son garantizadas para encontrar todos los datos relevantes dentro de esa memoria principal (2).

Periódicamente o cuando la base de datos requiera los datos se escriben en los archivos de punto de control (3).

Una aplicación COMMIT desencadena una escritura en el registro de transacciones (4).

Aunque por defecto esta escritura será asincrónica para que la aplicación no necesite esperar a escribir en disco. El registro de transsacciones puede usarse para recuperar la base de datos en un evento de falla (5).

TimesTen es importante en la actualidad, principalmente, como parte de la arquitectura de soluciones empresariales de Oracle. De todos modos, eso representa un buen ejemplo de una arquitectura de base de datos relacional transaccional basada en memoria temprana.

Redis

Si bien TimesTen es un intento de crear una base de datos en memoria compatible con RDBMS, Redis está en el extremo opuesto: ya que es esencialmente un almacén de clave-valor en memoria.

Redis (servidor de diccionario remoto) fue originalmente concebido como un simple sistema en memoria capaz de mantener tasas de transacción muy altas en sistemas de poco poder, como imágenes de máquinas virtuales.

Redis fue creado por Salvatore Sanfilippo en 2009. VMware contrató a Sanfilippo y patrocinó el desarrollo de Redis en 2010. En 2013, el software Pivotal, un spin-off de Big Data de la empresa matriz de VMware EMC-se convirtió en el principal patrocinador.

Actualmente, Redis es un almacén de estructura de datos en memoria de código abierto (con licencia de BSD, *https://redis.io*), que se utiliza como base de datos, caché y agente de mensajes. Admite estructuras de datos tales como cadenas, hashes, listas, conjuntos, conjuntos ordenados con consultas de rango, mapas de bits, hyperloglogs e índices geoespaciales con consultas radiales.

Redis tiene una replicación incorporada, secuencias de comandos Lua, desalojo LRU, transacciones y diferentes niveles de persistencia en disco, y proporciona alta disponibilidad a través de Redis Sentinel y particiones automáticas con Redis Cluster.

Puede ejecutar operaciones atómicas en estos tipos, como agregar a una cadena; incrementando el valor en un hash; empujar un elemento a una lista; computar o establecer una intersección, unión y diferencia; o conseguir el miembro con la clasificación más alta en un conjunto ordenado.

Para lograr su rendimiento sobresaliente, Redis trabaja con un conjunto de datos en memoria. Dependiendo de su caso de uso, puede persistir ya sea volcando el conjunto de datos en el disco de vez en cuando, o agregando cada comando a un registro.

La persistencia se puede inhabilitar opcionalmente, si solo necesita una memoria caché en la memoria, rica en funciones y en red. Redis también es compatible con la replicación asincrónica maestro-esclavo trivial a la configuración, con primera sincronización rápida sin bloqueo, autoconexión con resincronización parcial en la división de red. Otras características incluyen:

- Transacciones
- Pub / Sub
- Lua scripting
- Llaves con un tiempo de vida limitado
- Desplazamiento LRU de llaves
- Conmutación por falla automática

Redis se puede usar desde la mayoría de los lenguajes de programación que existen.

Redis está escrito en ANSI C y funciona en la mayoría de los sistemas POSIX como Linux, BSD, OS X sin dependencias externas. Linux y OS X son los dos sistemas operativos donde Redis está desarrollado y más probado, y se recomienda endamos usar Linux para implementar. Redis puede funcionar en sistemas derivados de Solaris como SmartOS, pero el soporte es el mejor esfuerzo. No hay soporte oficial para compilaciones de Windows, pero Microsoft desarrolla y mantiene un puerto Win-64 de Redis.

Aunque Redis fue diseñado para contener todos los datos en la memoria, es posible que Redis opere en conjuntos de datos memoria más grande que el disponible mediante el uso de su función de memoria virtual. Cuando esto esté habilitado, Redis "intercambiará por fuera" valores-clave más antiguos a un archivo de disco.

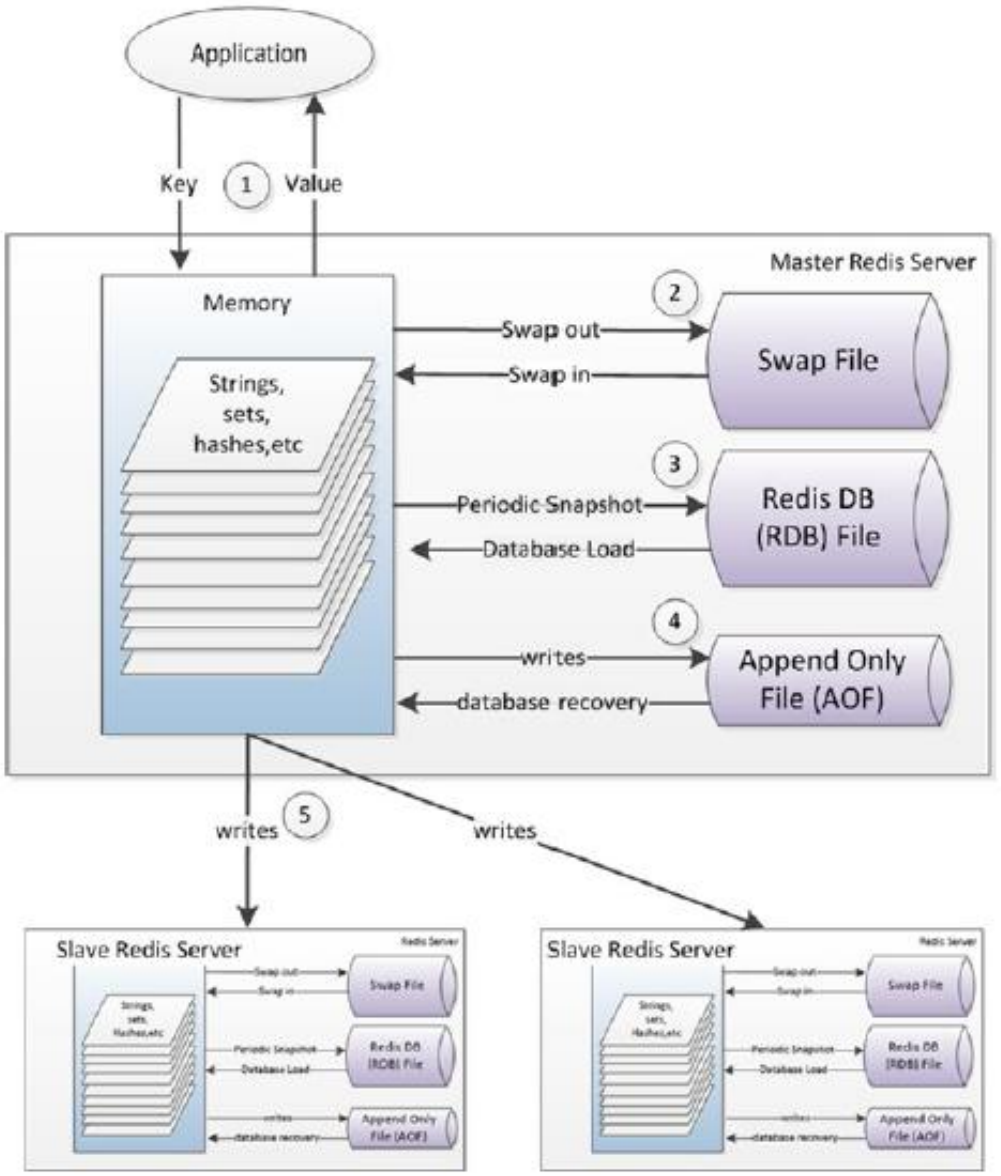
Si se necesitan las claves serán devueltas a la memoria. Esta opción obviamente implica una sobrecarga de rendimiento significativa, ya que algunas búsquedas declave darán como resultado un IO de disco.

Redis usa archivos de disco para persistencia:

- Los archivos de instantáneas almacenan copias de todo el sistema Redis en un momento determinado. Las instantáneas se pueden crear a pedido o se pueden configurar para que ocurran a intervalos en una hora programada intervalos o después de que se haya alcanzado un umbral de escritura. También se produce una instantánea cuando el servidor se apaga.
- El "archivo de apendar únicamente" (AOF, append only file) mantiene un diario de los cambios que se pueden utilizar para "ir hacia atrás" en la base de datos desde una instantánea en el caso de una falla. Las opciones de configuración permiten al usuario configurar escrituras en el AOF después de cada operación, en intervalos de un segundo o basados en intervalos de descarga determinados por el sistema operativo. Además, Redis admite la replicación principal/esclavo asincrónica.

Si el rendimiento es muy crítico y algunos datos perdidos son aceptables, luego se puede usar una réplica como base de datos de respaldo y el principal configurado con una mínima persistencia basada en disco. Sin embargo, no hay forma de limitar la cantidad de posible pérdida de datos; durante cargas altas, el esclavo puede caer significativamente detrás del principal.

La siguiente figura ilustra estos componentes arquitectónicos.



Guy Harrison, Next Generation Databases, 2016

La aplicación interactúa con Redis a través de búsquedas de claves principales que devuelven "valores": cadenas, conjuntos de cadenas, hash de cadenas, etc. (1).

Los valores clave casi siempre estarán en la memoria, aunque es posible configurar Redis con un sistema de memoria virtual, en cuyo caso los valores de clave pueden tener que ser intercambiados dentro o fuera (2).

Periódicamente, Redis puede volcar una copia de todo el espacio de memoria en el disco (3).

Además, Redis se puede configurar para escribir cambios en un AOF diario solo a intervalos cortos o después de cada operación (4).

Finalmente, Redis puede replicar el estado de la base de datos maestra de forma asíncrona a los servidores Redis esclavos (5).

Aunque Redis se diseñó desde cero como en un sistema de base de datos en memoria, las aplicaciones puede que tenga que esperar a que el IO lo complete en las siguientes circunstancias:

- Si el archivo de solo agregar está configurado para escribirse después de cada operación, entonces la la aplicación necesita esperar a que se complete un IO antes de que vuelva una modificación de control.
- Si se configura la memoria virtual Redis, entonces la aplicación puede necesitar esperar una clave para ser "intercambiado" por la memoria.

Redis es popular entre los desarrolladores como un DW simple, de alto rendimiento y de tipo clave-valor que trabajabien sin hardware costoso.

Carece de la sofisticación de algunos otros sistemas no relacionales como MongoDB, pero funciona bien en sistemas donde los datos caben en la memoria principal o como una capa de almacenamiento en caché en frente a una base de datos basada en disco.

SAP HANA

SAP presentó HANA en 2010, posicionándolo como una revolucionaria base de datos en memoria diseñada principalmente para Business Intelligence (BI), pero también capaz de soportar cargas de trabajo OLTP (*https://www.sap.com/latinamerica/products/hana.html*).

SAP HANA es una base de datos relacional diseñada para proporcionar un rendimiento innovador combinando tecnología en memoria con una opción de almacenamiento en columna, instalada en una configuración optimizada de hardware.

Aunque SAP no envía hardware de HANA, proporcionan directrices detalladas para servidores certificados por HANA, incluyendo un requisito para discos SSD rápidos. Las tablas en HANA pueden configurarse para almacenamiento en filas o columnas. Por lo general, las tablas destinadas a los propósitos de BI se configurarán como columnas, mientras que las tablas OLTP se configurarán como orientadas a filas.

La elección de formatos de fila o columna le proporciona a HANA una capacidad para proporcionar soporte tanto OLTP como analítico en cargas de trabajo. Se garantiza que los datos en el almacén de filas estarán en la memoria, mientras que los datos en el almacén de columnas están cargados por defecto bajo demanda.

Sin embargo, columnas especificadas o tablas completas pueden configurarse para carga inmediata en el inicio de la base de datos. La arquitectura de persistencia de HANA utiliza el patrón de archivo de instantánea y diario que se encuentra en Redis y TimesTen.

HANA captura instantáneamente el estado de la memoria en archivos Savepoint. Estos Savepoints son aplicados periódicamente a los archivos maestros de la base de datos. La consistencia transaccional de ACID se habilita mediante el registro de transacción "rehacer".

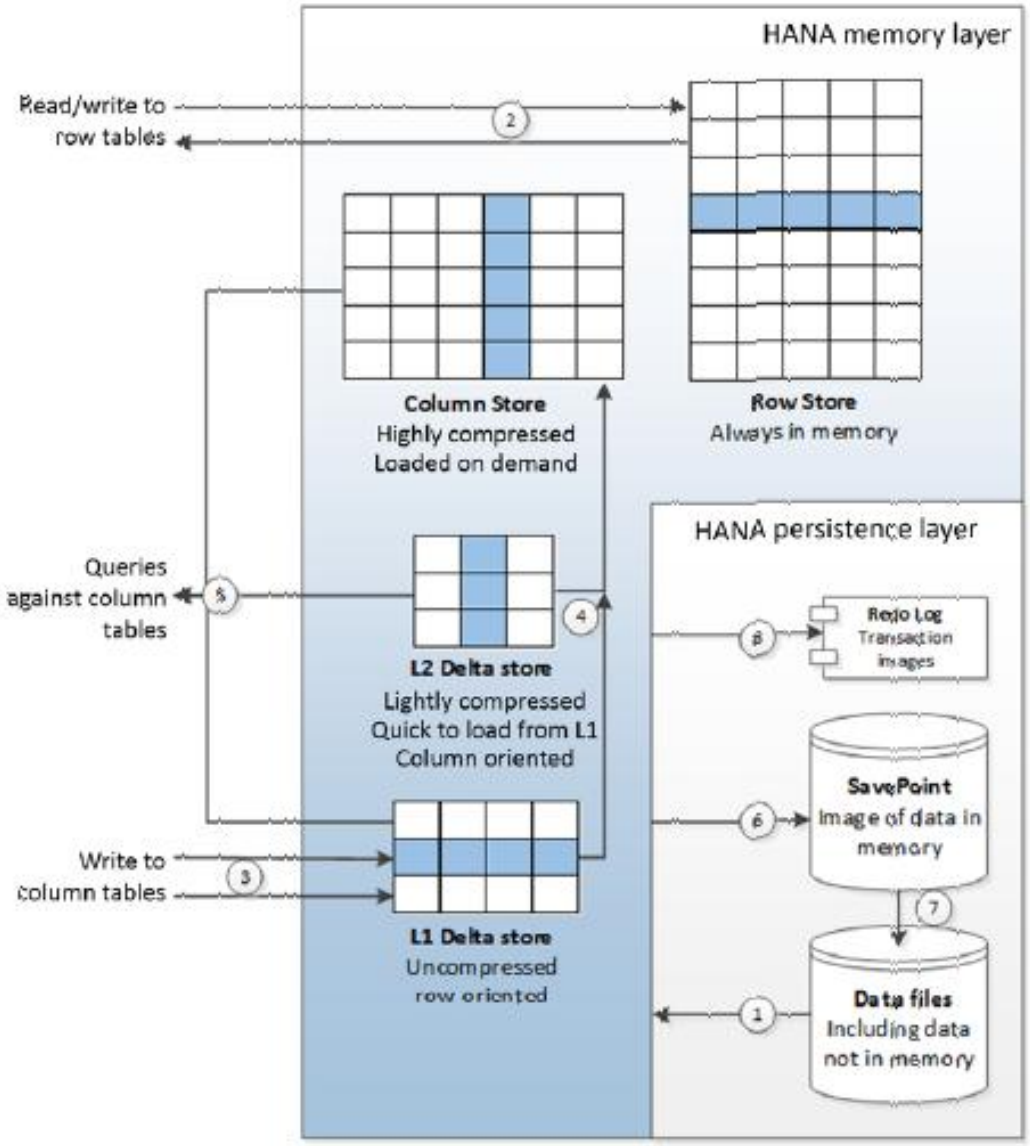
Como con la mayoría de las bases de datos relacionales que obedecen ACID, este registro se escribe al confirmar la transacción, lo que significa que las aplicaciones esperarán el registro de transacciones IO para completarse antes que la confirmación puede devolver el control.

Para minimizar las esperas de IO, que podría ralentizar las operaciones de HANA de velocidad de memoria, el registro de rehacer se coloca en dispositivos de disco de estado sólido HANA certificados por SAP.

La arquitectura de columna de HANA incluye una implementación del patrón de almacén delta optimizado para escritura. Las transacciones a tablas columnas están almacenadas en esta tienda delta. Inicialmente, los datos se mantiene en formato orientado a filas (el delta L1).

Los datos luego se mueven a la tienda L2 delta, que es de columnas en orientación, pero relativamente comprimido y sin clasificar.

Finalmente, los datos migran a la columna principal del almacén, en el que los datos están altamente comprimidos y ordenados. La siguiente figura muestra estos aspectos clave de la arquitectura de HANA.



Guy Harrison, Next Generation Databases, 2016

En la puesta en marcha, tablas basadas en filas y tablas de columnas seleccionadas se cargan desde los archivos de la base de datos (1).

Se cargarán otras tablas de columnas demandadas. Las lecturas y escrituras en tablas orientadas a filas se pueden aplicar directamente (2).

Actualizaciones de las tablas basadas en columnas se almacenarán en el almacén delta (3),

Inicialmente en el almacén orientado a filas L1. Los datos en la tienda L1 en consecuencia, se promociona a la tienda L2, y finalmente a la tienda de columnas en sí (4).

Consultas con las tablas columna se deben leer tanto de la tienda delta como del almacén de columnas principal (5).

VoltDB

Redis, HANA y TimesTen son sistemas legítimos de bases de datos en memoria. diseñados desde cero para usar la memoria como la fuente primaria, y generalmente exclusiva, para todos los datos.

Sin embargo, como hemos visto, las aplicaciones que usan estos sistemas a menudo necesitan esperar el IO del disco. En particular, a menudo habrá un disco IO a algún tipo de diario o registro de transacciones cuando se compromete una transacción.

VoltDB es una implementación comercial del diseño de H-store (*https://www.voltdb.com*). H-store describe una base de datos en memoria diseñada con la intención de no requerir IO de disco durante las operaciones transaccionales normales: aspira a ser una solución de memoria pura.

VoltDB admite el modelo transaccional ACID, pero en lugar de garantizar la persistencia de los datos mediante la escritura en un disco, la persistencia está garantizada mediante la replicación en varias máquinas.

Una confirmación de transacción solo se completa una vez que los datos se escriben correctamente en la memoria en más de una máquina física. La cantidad de máquinas involucradas depende del nivel de seguridad K especificado.

Por ejemplo, un nivel de seguridad K de 2 no garantiza la pérdida de datos si fallan dos máquinas; en este caso, la confirmación debe propagarse con éxito a tres máquinas antes de completar.

También es posible configurar un registro de comando que registre los comandos de transacción en un registro basado en disco. Este registro puede escribirse sincrónicamente en cada proceso de transacción o de forma asíncrona.

Porque VoltDB solo admite transacciones basadas en procedimientos almacenados deterministas, el registro solo necesita registrar el comando real que condujo la transacción, en lugar de copias de bloques modificados, como es común en otros registros de transacciones de bases de datos.

No obstante, si se utiliza la opción de registro de comandos síncronos, entonces las aplicaciones de VoltDB pueden experimentar el tiempo de espera de IO de disco.

VoltDB admite el modelo relacional, aunque su esquema de agrupamiento funciona mejor si ese modelo puede ser particionados jerárquicamente a través de claves comunes. En

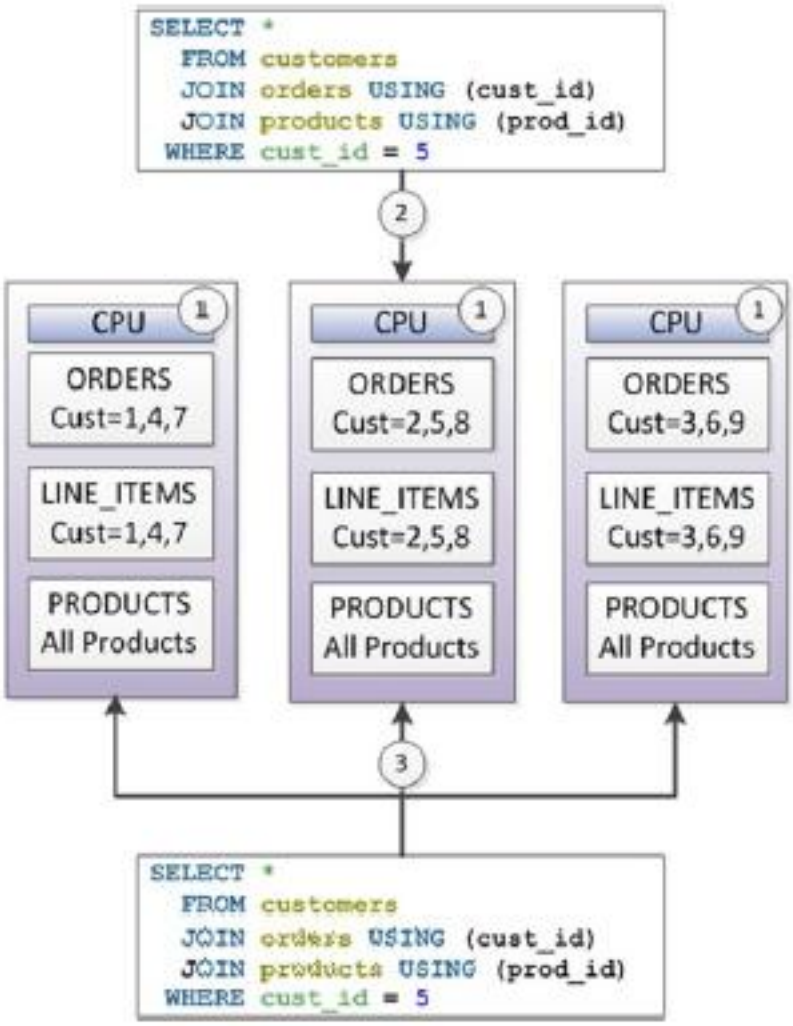
VoltDB, las tablas deben ser particionadas o replicadas. Las particiones se distribuyen entre los nodos del clúster, mientras que las tablas replicadas se duplican en cada partición.

Las tablas que se replican pueden incurrir en gastos indirectos adicionales durante las operaciones OLTP, ya que las transacciones deben estar duplicadas en cada partición; esto está en contraste con las tablas divididas, que incurren en sobrecarga si los datos deben recopilarse desde múltiples particiones.

Por esta razón, las tablas replicadas son usualmente tablas de tipo de referencia más pequeñas, mientras que las tablas transaccionales más grandes están particionadas. Las particiones en VoltDB se distribuyen no solo a través de máquinas físicas sino también dentro de máquinas que deben tener múltiples núcleos de CPU.

En VoltDB, cada partición está dedicada a una única CPU que tiene exclusividad de acceso de un solo subproceso a esa partición. Este acceso exclusivo reduce la sobrecarga de bloqueo y enganche, pero sí requiere que las transacciones se completen rápidamente para evitar la serialización de las solicitudes.

La siguiente figura muestra cómo la partición y la replicación afectan la concurrencia en VoltDB.



Guy Harrison, Next Generation Databases, 2016

Cada partición es asociado con un único núcleo de CPU, y solo una sentencia de SQL estará activa contra la partición en cualquier momento dado (1).

En este ejemplo, ORDERS y CUSTOMERS se han dividido utilizando el id de cliente, mientras que la tabla PRODUCTS se replica en su totalidad en todas las particiones. Una consulta que accede a datos para las necesidades de un único cliente involucran solo una partición (2),

Mientras que una consulta que funciona entre los clientes debe tener acceso-exclusivo, aunque por un tiempo corto a todas las particiones (3).

Las transacciones de VoltDB se simplifican al ser encapsuladas en una sola llamada de procedimiento almacenado de Java, en lugar de estar representado por una colección de declaraciones SQL separadas.

Esto asegura que las duraciones de transacción se reducen al mínimo (sin tiempo de reflexión o tiempo de red dentro de las transacciones) y reduce aún más las cuestiones de bloqueo.

Bibliografía utilizada y sugerida

Catherine M. Ricardo, Susan D. Urban, Databases Illuminated, USA New York, Jones & Bartlett Learning 2017.

Guy Harrison, Next Generation Databases, USA New York, APRESS 2016.

Lo que vimos:

En esta Unidad vimos los conceptos de bases de datos en memoria, sus ventajas y desventajas. Abordamos, también, el conocimiento de las distintas herramientas existentes en el mercado.

Lo que viene:

En la Unidad siguiente vamos a ver de qué tratan las bases de datos orientadas a objetos, qué ventajas ofrecen y qué desafíos presentan a administradores y desarrolladores.