

## SISTEMAS DE BASES DE DATOS

### KEY-VALUE STORES

#### DYNAMO

Prof. María Constanza Pabón

Dynamo, Amazon

DeCandia et al. Dynamo: Amazon's Highly Available Key-value Store. SOSP, 2007

Amazon tiene una plataforma para comercio electrónico mundial

Sirve a decenas de millones de clientes

Se compone de cientos de servicios que trabajan coordinadamente para entregar la funcionalidad

Recomendaciones, ordenes de compra, detección de fraude, ...

Consta de miles de servidores ubicados en varios centros de datos alrededor del mundo

Algunos son stateless, otros stateful

Las aplicaciones tienen diferentes requerimientos de almacenamiento

Arquitectura altamente descentralizada, con bajo acoplamiento, orientada a servicios

Dynamo: Requerimientos

Varios servicios, entre ellos las listas de los más vendidos, carritos de compras, preferencias de los clientes, gestión de sesiones, ranking de ventas, y catálogos de productos, requieren:

Modelo de consulta

Escritura y lectura de ítems de datos que se identifican por una llave

Objetos binarios (blobs) identificados por llaves únicas

Las operaciones no involucran varios ítems de datos

Objetos relativamente pequeños ( < 1 MB )

Dynamo: Requerimientos

Alta disponibilidad, consistencia débil

El mas pequeño corte de servicio tiene consecuencias financieras amplias e impacta la fidelidad del cliente

Con miles de servidores siempre hay algunos fallando (es lo normal)

Ejemplo: Los clientes deben poder ver y agregar ítems a su carrito de compras aunque los discos fallen, las redes presenten problemas, o los centros de datos se destruyan

Alta escalabilidad

Crecimiento continuo de la plataforma

Lógica de negocio simple

Dynamo: Requerimientos

Desempeño, eficiencia

Funciona sobre servidores de bajo costo

99.9% de cumplimiento de los SLAs (Service Level Agreement)

Ejemplo SLA: entregar respuesta en menos de 300ms para cargas de 500 requerimientos por segundo

Un requerimiento de una página puede necesitar enviar una solicitud a 150 servicios, que a su vez tienen múltiples dependencias (grafo)

No usan medidas como promedio o mediana porque ocultan casos de clientes que reciben mal servicio

Un mayor porcentaje de cumplimiento involucra un alto incremento en costo, sin mejorar el desempeño ampliamente

Dynamo: Requerimientos

Otras características:

Los servicios son internos

No operan en un ambiente hostil

No hay requerimientos de seguridad de acceso (autenticación, autorización)

Tradeoffs:

desempeño, eficiencia en costo, disponibilidad, durabilidad

Dynamo: Consideraciones de diseño

La disponibilidad y la escalabilidad dependen de como se maneja el estado

Los sistemas de almacenamiento afectan ampliamente los SLA cuando la lógica de negocio es simple

Los SGBD tradicionales sincronizan las réplicas para proveer consistencia fuerte

Afectan la disponibilidad

No se permite incertidumbre: se bloquea el acceso hasta que se tiene certeza de que el dato es correcto

Replicación optimista vs Replicación pesimista

Las técnicas optimistas de replicación pueden llevar a conflictos que deben ser detectados y resueltos:

Cuando y Quien

Dynamo: Consideraciones de diseño

Cuando resolver conflictos de actualización

En la escritura:

Puede requerir rechazar operaciones de escritura

En la lectura:

Dynamo es un almacenamiento "always writable": muchos servicios no rechazan una actualización

Ej. Carrito de compras

Quien resuelve los conflictos de actualización

El data store

Políticas simples y genéricas: ej. "last write wins"

La aplicación

Políticas más complejas, adaptadas a la semántica de los datos

Dynamo: Consideraciones de diseño

Simetría

Todo nodo tiene las mismas responsabilidades que sus pares

Descentralización

Aplicación de técnicas peer-to-peer

Heterogeneidad

Explotar la heterogeneidad de los nodos (infraestructura)

Dynamo: Componentes

Otros:

Persistencia

Balanceo de carga

Manejo de sobrecargas

Transferencia del estado

Programación de trabajos y de la concurrencia

Marshalling

Ruteo de solicitudes

Monitoreo del sistema y alarmas

Administración de la configuración

Core:

Particionamiento (fragmentación)

Replicación

Manejo de fallos (detección y recuperación)

Membresía (de los nodos)

Escalamiento

Dynamo: Operaciones internas

Get(key): localiza las réplicas del objeto asociado con la clave dada y retorna un solo objeto o una lista de objetos (versiones en conflicto) con su contexto

Contexto: metadatos de sistema, ej. la versión del objeto

Put(key, context, object): determina donde guardar las réplicas del objeto (basado en la clave) y las escribe en disco

La llave y el objeto se manejan como un arreglo de bytes opaco

Aplica MD5 en la clave para generar un identificador de 128 bits, que se usa para determinar los nodos de almacenamiento

Dynamo: Partición dinámica

Consistent Hashing: modificado para asegurar una distribución uniforme de los datos y aprovechar la heterogeneidad de las máquinas

Consistent Hashing

Las particiones mapean rangos de key hashes (función hash aplicada a la llave)

El máximo valor del hash es  $2^{160}$

Por defecto se tienen 64 particiones

Cada nodo físico se compone de nodos virtuales, a los que se asignan las particiones

Cada objeto se replica en las n particiones siguientes

De acuerdo con el número de réplicas definidas para el bucket

Ventajas de los nodos virtuales

Permite repartir uniformemente la carga de un nodo (físico) que deja de estar disponible (por fallas o mantenimiento)

Un nodo que ingresa al sistema (o que vuelve a estar disponible) recibe una carga equivalente a la de los otros nodos

El número de nodos virtuales que recibe un nodo se decide con base en su capacidad, de esta forma se administra la heterogeneidad de la infraestructura física

Dynamo: Replicación

Cada ítem de datos es replicado en N nodos (N es configurable)

Un dato se replica en múltiples centros de datos

Cada clave, k, se asigna a un nodo coordinador que está a cargo de la replicación en los N-1 nodos siguientes en el anillo

Se asegura que no sea el mismo nodo físico (consistent hashing modificado)

Dynamo: Consistencia eventual

Permite propagar actualizaciones de forma asíncrona

Permite configurar:

W: mínimo número de nodos que deben participar en una operación de escritura exitosa

R: mínimo número de nodos que debe participar en una operación en una de lectura exitosa

Quorum approach:  $N=3$ ,  $W=2$ ,  $R=2$  ( $W+R>N$ )

Los valores N, W, R impactan la disponibilidad, consistencia y durabilidad

Sloppy Quorum (Quorum Descuidado), Hinted Handoff

Las operaciones se ejecutan en los primeros M nodos saludables

Cuando un nodo no responde, otro nodo guarda los cambios en una bd local separada hasta que el nodo regrese o se detecte un fallo permanente

Dynamo: Manejo de versiones

Cada escritura (o actualización) crea un nuevo objeto inmutable

En su experiencia, múltiples versiones de un dato surgen por dos razones:

Fallas del sistema

Gran número de escrituras concurrentes

Si la versión más reciente del dato no esta disponible para actualización o borrado se actualiza una versión anterior

Las diferentes versiones se reconcilian después

Dynamo: Manejo de versiones

Dynamo hace reconciliación sintáctica

Cuando un vector clock contiene al otro

La reconciliación semántica (cuando hay varias ramas) se programa en la aplicación

Ej. En el carrito de compras, se asegura que un "add to cart" nunca se pierda; pero un "delete from cart" se puede perder (merge).

Dynamo: Arquitectura

Para servicios que requieren mejor desempeño las actualizaciones se mantienen en un buffer en memoria

Periódicamente se escribe el buffer en el almacenamiento local

Un fallo en el servidor puede ocasionar perdida de datos

El riesgo se minimiza porque se tienen N nodos con los datos replicados

Dynamo: Implementación

Cada nodo tiene 3 componentes (implementados en Java)

Coordinador de requerimientos

Membresía y detección de fallos

Motor de persistencia local

Permite el uso de diferentes motores: Berkeley DB (BDB), BDB Java Edition, MySQL

El motor de persistencia se elije de acuerdo con el tamaño de los objetos a almacenar

Ej. MySQL es mejor para objetos grandes

Referencias

DeCandia et al. Dynamo: Amazon's Highly Available Key-value Store. SOSP, 2007