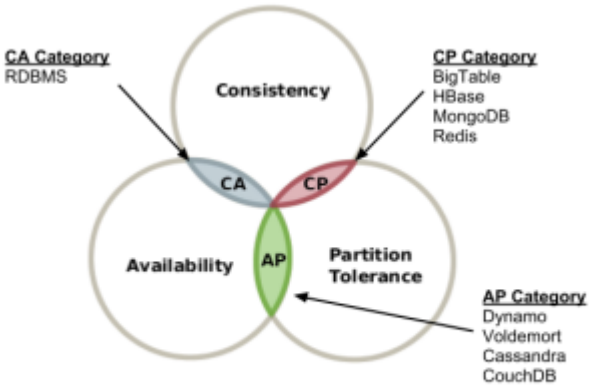
-Cassandra-

Almacenamiento de datos distribuido

En los últimos años, con la necesidad de manejar grandes volúmenes de datos, se han implementado diferentes formas de afrontar este reto. En el presente artículo, se abordan conceptos básicos de Cassandra, un sistema de almacenamiento de datos para Big Data.

Un concepto importante antes de continuar es el teorema CAP (Consistency, Availability, Partition Tolerance), dicho teorema dice que en un sistema gestor de bases de datos solo podremos tener 2 de las 3 categorías (Consistencia, Disponibilidad, Tolerancia a fallos). Esto no es ni bueno ni malo, pero debe ser considerado según el área de aplicación y el problema a resolver.

Cassandra se ubica en AP (alta disponibilidad y tolerancia fallos, sacrificando consistencia). Cuando se aborte la arquitectura, quedará más claro el concepto.



Teorema CAP

HISTORIA

Cassandra surge como una solución de Facebook para resolver el acceso al inbox (mensajería interna de la red social). Básicamente para responder en tiempo real consultas a los datos almacenados, considerando los grandes volúmenes de datos, usuarios y consultas.

Se basa en DynamoDB (solución de Amazon para tratar grandes volúmenes de datos) y Bigtable (Google)

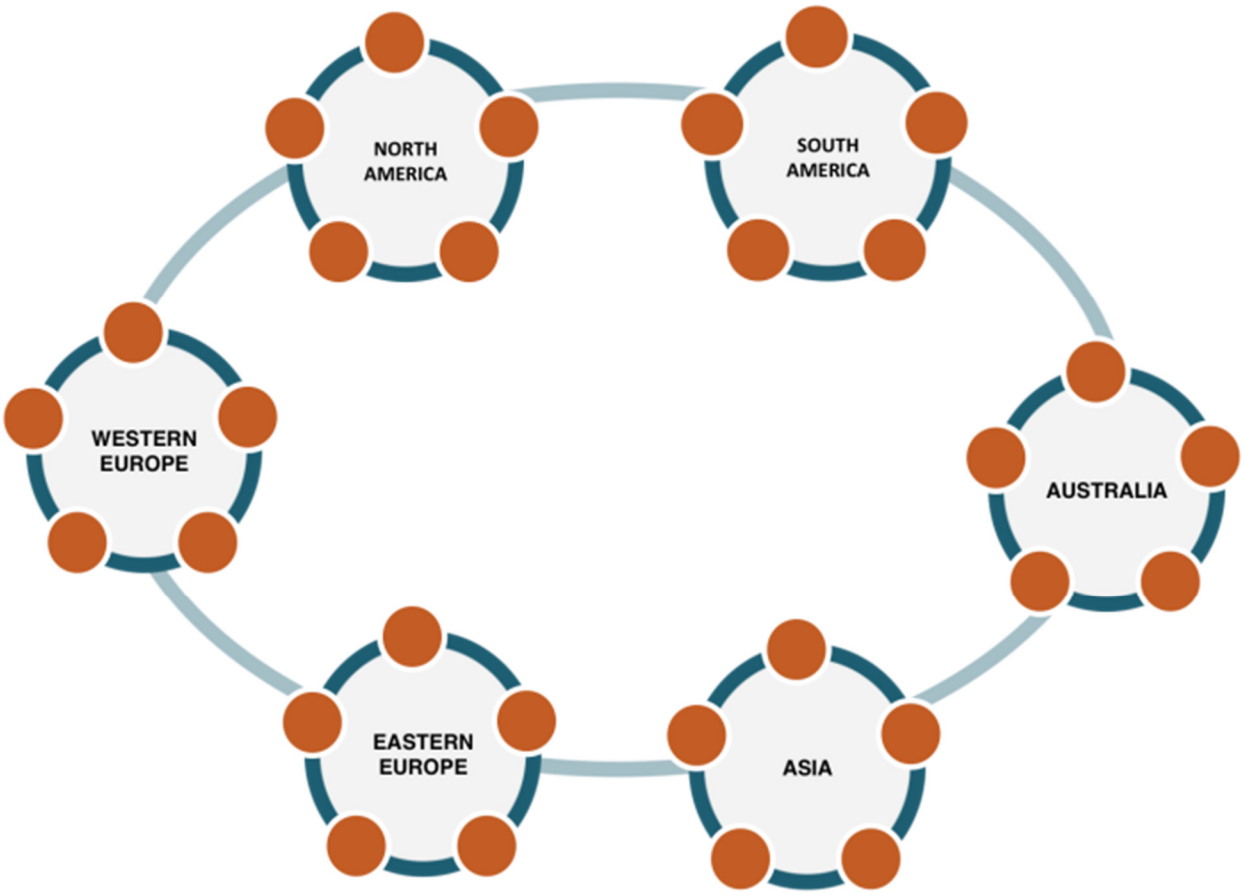
Es una Base de Datos NoSQL (Not Only SQL) que puede manejar datos estructurados, semi-estructurados y no estructurados

---

ARQUITECTURA

Es un sistema distribuido, es decir los datos se distribuyen en nodos, y todos los nodos “son iguales”, es un modelo peer-to-peer, donde no existe el concepto de master-slave; existen roles de coordinador, nodo primario, nodos réplica, pero cualquier nodo puede tomar cualquier rol en distintas consultas. Esto es una diferencia importante respecto a otras soluciones conocidas como Google File System (GFS).

Un cluster está conformado por N nodos, y estos están pueden estar agrupados en N DataCenter

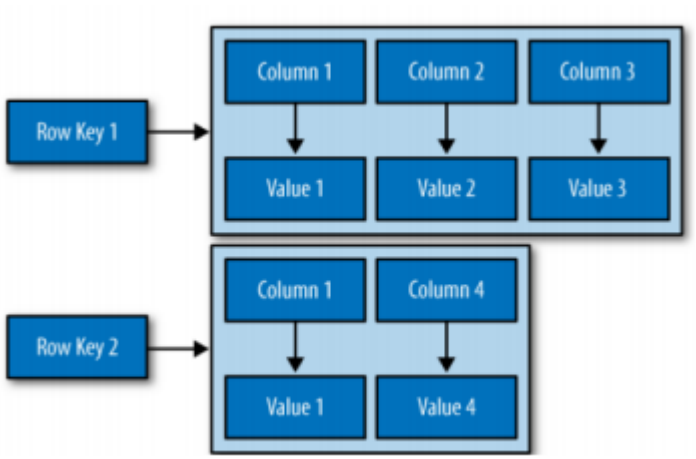


Nodos en DataCenter

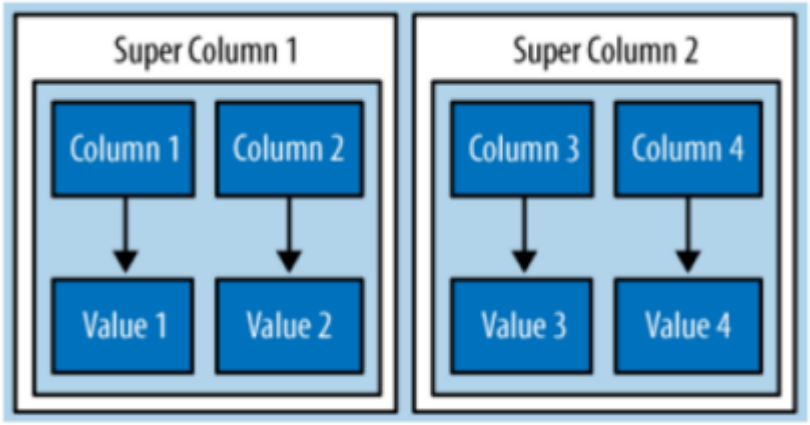
Combina características de una base de datos clave-valor (como Bigtable) y de las bases de datos relacionales con el manejo de columnas, familias de columnas y supercolumnas. En Bigtable conocimos el concepto de familias de columnas con lo que se pueden tener hasta 3 niveles de profundidad (dato, columna, familia), en Cassandra al incluir el concepto de super-columna y

familias de supercolumnas (básicamente familias de familias), se pueden tener múltiples niveles, agrupando columnas en super-columnas.

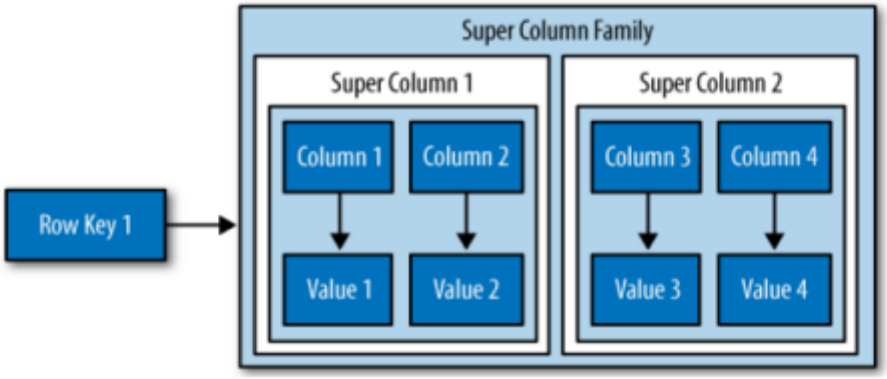
Al igual que Bigtable y GFS, Cassandra usa timestamp para manejo de versiones y cuenta con un recolector de basura para desechar datos por antigüedad



Fila: columna-valor



Super Columna: familias



de columnas
Familias de super columnas: familias de familias

Ventajas:

- No existe un único punto de fallo (master).
- Alta disponibilidad, y tolerancia a fallos si un nodo falla el sistema continúa operando
- No existen cuellos de botella
- Escalabilidad lineal. Se incrementa la capacidad linealmente al aumentar la cantidad de nodos

Desventajas:

- Se reduce el rendimiento al agregar nuevos niveles (familias de super-columnas)
- Según el teorema CAP, al tener alta disponibilidad y tolerancia a fallos, no se puede asegurar la consistencia.
- Sin embargo, en Cassandra el nivel de consistencia es configurable, básicamente indicando en cuantos nodos tener el mismo dato para darlo por bueno. A mayor consistencia, menor rendimiento y menor velocidad de respuesta.

PROTOCOLO DE MENSAJES

Cassandra usa Gossip como protocolo de comunicación para propagar datos en el sistema, para intercambiar información entre nodos como conocer otros nodos y el estado del sistema.

Conceptos básicos del protocolo Gossip:

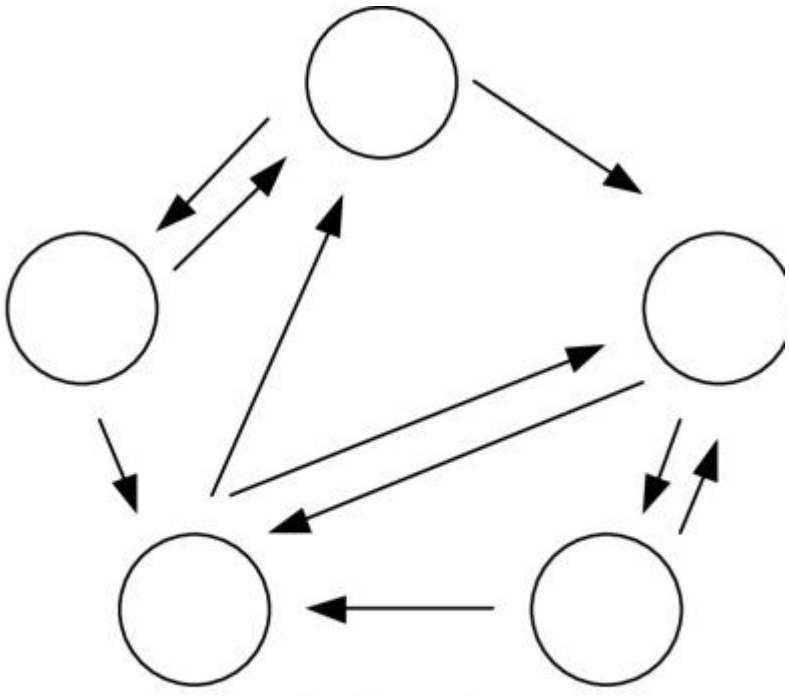
- Operación “querie” (consultas de solo lectura)
- Operación “update” (modifican el estado, sin leerlo antes, sobreescriben lo que hay)
- TimeStamps: para manejo de consistencia, versiones, y propagación de estados
- Ciclos: parámetro que define la cantidad de repeticiones para distribuir un mensaje
- Fanout: parámetro que define cantidad de nodos alcanzados en cada ciclo

Funcionamiento de Gossip:

El potencial de Gossip radica en su sencillez, y en estar enfocado para usar en un modelo descentralizado, a alto nivel, funciona de la siguiente manera:

- 1. Un nodo quiere enviar un mensaje a otros nodos en la red
- 2. Selecciona N nodos aleatorios para enviar el mensaje (N es el fanout)
- 3. Repite la misma operación tantas veces como ciclos fueron configurados
- 4. Cada nodo que recibe el mensaje repite los pasos 2 y 3

Para enviar un mensaje a todos los nodos, la cantidad de ciclos crece logarítmicamente respecto a la cantidad de nodos. En un sistema de BigData con grandes cantidades de nodos, esto puede convertirse en un problema dependiendo del volumen. Otra desventaja en el uso de Gossip es que cada nodo recibe el mensaje, y agrega información de sí mismo para enviarlo a otro nodo, esto hace que el tamaño del mensaje crezca con cada nodo.



(c) Gossip-based approach, where peers operate in parallel, and each peer communicates with one or more randomly selected partner

5 nodos, fanout = 2

MARCO APLICATIVO

La aplicación de uno u otro sistema dependerá de la naturaleza de los datos y la combinación más adecuada de Consistencia, Tolerancia a fallos y alta disponibilidad.

Cassandra al no asegurar consistencia, posiblemente no sea aplicable en sistemas transaccionales o en la operación diaria de un negocio de este tipo; pero de todos modos tampoco es este el sentido del BigData. Mientras que sí podría ser utilizado en sistemas CRM, análisis de datos generados en redes sociales, interacciones de clientes, Internet de las Cosas, etc.

Se desarrolló a lo interno de Facebook, y posteriormente pasó a ser Open Source, con lo que otras compañías lo aplican actualmente, en una serie de usos diarios tales como:

· Análisis de datos de redes sociales

· Búsqueda en catálogos de tiendas on-line

· Datos de series en el tiempo, tales como datos de clima

· Histórico de conversaciones en aplicaciones de mensajería

· Rastreo y monitoreo de preferencias y actividades de usuario en páginas web, aplicaciones de streaming, etc.

Grandes compañías hacen uso de Cassandra, tales como Facebook, Twitter, Netflix, Apple, Google, Amazon, eBay, Microsoft, IBM, entre otras