Introducción

Apache Cassandra - Bases de datos II

Alberto Díaz Álvarez (<alberto.díaz@upm.es>)
Departamento de Sistemas Informáticos
Escuela Técnica superior de Ingeniería de Sistemas Informáticos

En este tema hablaremos de las bases de datos clave-valor

- Sus propiedades, escalabilidad, indexación, etcétera
- Nos centraremos en Apache Cassandra, aunque sea un modelo clave-valor híbrido
  - Híbrido porque incluye el concepto de columnas

Comenzamos

Bases de datos clave-valor

¿Qué es una base de datos clave-valor?

Es un sistema que almacena valores indexados por claves

- Vamos, como una tabla *hash*, pero a lo bestia
- Permite almacenar datos estructurados y no estructurados

Sobre Apache Cassandra

Sistema de gestión de bases de datos NoSQL distribuido

- Arrancado por Facebook y posteriormente cedido a la Apache Foundation
- Gratuito y de código abierto (licenciado bajo la Apache License 2.0)
- Desarrollado en el lenguaje de programación Java y multiplataforma
- Diseñado para manejar grandes cantidades de datos entre muchos nodos
- Soporta clústeres distribuidos en diferentes clústeres
- Además con replicación asíncrona sin nodo maestro
- Esto permite operaciones de baja latencia
- Web: https://cassandra.apache.org

Algunas ventajas de cassandra

Está desarrollado directamente para ser un servidor distribuido y descentralizado

- Pero permite ser ejecutado como nodo simple
- De cara al cliente, es transparente y siempre se ofrece como un servicio único

Lectura y escritura rápida sin afectar al servicio

- Ofrece administración de volúmenes de datos distribuidos e incrementales

Escalabilidad horizonal, permitiendo añadir nuevo hardware bajo demanda

- Alta disponibilidad sin punto único de fallo (SPOF, de *single point of failure*)
- Básicamente todo nodo del clúster conoce al resto

API simple para el acceso desde cualquier lenguaje de programación

Y algunas desventajas, claro

Las consultas sobre el sistema son *ad hoc*

- Se modela sobre las consultas a realizar, no sobre la estructura de los datos

No soporta agregaciones como parte del modelo subyacente

- No usan el álgebra relacional, donde las agregaciones son naturales
- Esto es, las operaciones (AVG, MAX, MIN, SUM) son **MUY** costosas

Rendimiento impredecible

- Ejecuta por debajo muchos procesos asíncronos y tareas de segundo plano
- Esto se traduce en un rendimiento impredecible

Protocolos de comunicación entre nodos

Cassandra utiliza un mecanismo de comunicación denominado *gossip* (cotilleo)

- Comunicación interna para permitir la comunicación dentro de un anillo
- De esta manera, cada nodo conoce a otros nodos
- Este protocolo ayuda a la descentralización y a la tolerancia a fallo

La estructura más externa de Cassandra es un clúster o anillo

- Un clúster está formado por nodos
- Cada uno de ellos contiene una réplica para diferentes rangos de datos
  - Parámetro `replication_factor` al crear un `KeySpace`
  - Ya veremos cómo funciona más adelante
- Así, si algo sale mal, una réplica puede responder a las peticiones

Sobre el consistencia, la disponibilidad y la tolerancia a fallos

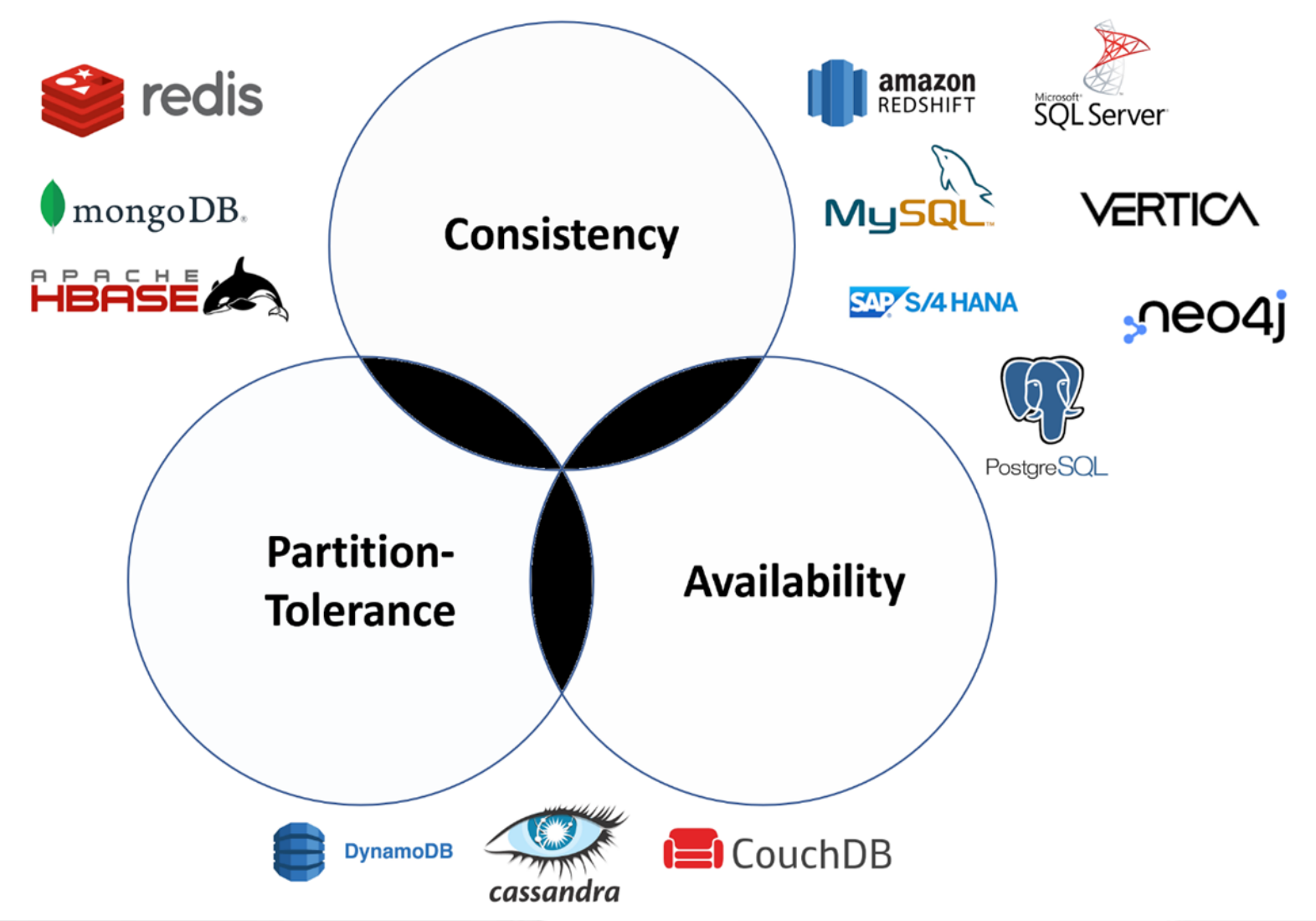
Teorema de Brewer

Establece que un sistema sólo puede garantizar a la vez dos requisitos entre:

- **Consistencia**: Toda lectura recibe la escritura más reciente o un error.
- **Disponibilidad**: Toda solicitud recibe una respuesta (no error)
- **Tolerancia al particionado**: Funcionamiento a pesar de pérdida de mensajes

Conocido también como **Teorema CAP** (***C****onsistency*, ***A****vailability*, ***P****artition tolerance*)

Teorema de Brewer



Teorema de Brewer y Cassandra

Desde un punto de vista práctico, la tolerancia a la partición es necesaria

- Los fallos en red son una constante en nuestra profesión
- Por ello las bases de datos suelen ser CP o AP

Cassandra es un sistema AP, es decir, prioriza disponibilidad sobre consistencia

- Esto es simplificar mucho, en realidad busca satisfacer los tres requisitos
- De hecho se puede configurar para que se comporte muy parecido a CP

Según la Apache Software Foundation, ofrece una **consistencia ajustable** de ms.

Garantizando la disponibilidad de los datos

La disponibilidad se garantiza a través de las réplicas

- Cuando se escribe un dato, se escribe en varias réplicas
- Generalmente tres, aunque este parámetro es configurable
- Así nos aseguramos de que si cae un nodo los datos no se pierden

Las réplicas demoran la escritura

- Hay momentos en los que diferentes nodos tienen diferentes valores
- Casandra se describe como **eventualmente consistente** por esto mismo

Es decir, se garantiza la disponibilidad de lso datos, pero no a su última versión

Consistencia "ajustable"

Cassandra permite "ajustar" el nivel N de consistencia (incluso por operación)

- $N \rightarrow$ **¿Cuántas réplicas** deben responder para completar una operación?

Por ejemplo, supongamos que queremos leer datos en un clúster de tres nodos

- El nivel **ONE** devolverá el valor de la **primera** réplica que responda
- El nivel **TWO** devolverá el valor de las **primeras dos** réplicas que respondan
- El nivel **THREE** o QUORUM esperará a que **todas** las réplicas respondan
- ¿Inconsistencias entre réplicas? Cassandra las gestiona antes
- **CUIDADO**: Si no se puede escribir en todas las réplicas, la operación fallará
- **CUIDADO**: Una mayor consistencia afectará significativamente al rendimiento

El **mismo proceso** se aplica a las **operaciones de escritura**

Primeros conceptos de Cassandra

Columna (*column*)

Es la unidad básica de almacenamiento

- Es una tupla que contiene un nombre, un valor y un *timestamp*
- Se pueden agrupar en súpercolumnas, familias de columnas y *keyspaces*
- Son análogas a las columnas de una tabla en el modelo relacional

El *timestamp* indica el momento de la última actualización de la columna

Súpercolumna (*super column*)

Es un array ordenado de columnas

- Se compone de una clave (*row key*), un nombre y su conjunto de columnas

Familia de columnas (*column family*)

Clúster

Conjunto de nodos que componen una instancia de Cassandra

Keyspace

Espacio de nombres para un conjunto de ColumnFamily, asociado a una aplicación

Gracias