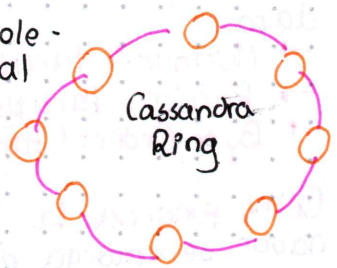
Cassandra

- Almacen altamente escalable, eventualmente consistente y distribuido de estructuras clave-valor y columnar.
- A través de clave-valor uno accede a las columnas, que a su vez son una tupla.
- Escrito en Java. Multiplataforma.
- Influido por Google BigTable y Amazon Dynamo.

ARQUITECTURA

- Distribuida: Los datos están (y servicios) en distintas máquinas e incluso distintos lugares.
- Escala linealmente: El rendimiento de procesamiento aumenta proporcionalmente al número de nodos que agrego.
- Arquitectura peer-to-peer (P2P): Ninguno de los nodos es más importante que el otro, y se conectan con todos. Cada nodo es una máquina que tiene una instancia de Cassandra funcionando.
- Escalabilidad horizontal:
- El cliente no determina a qué nodo conectarse. Un nodo va a contestar por algún criterio como distancia, latencia. El nodo que responde es el nodo coordinador, quien orquesta dónde quedaran los datos o donde los irá a buscar.
- Factor de replicación (FR): Los datos están repetidos en FR nodos. Es importante replicar en caso de que alguno falle (ej: se cae la red). Se configura al momento de crear la base de datos.
- Cluster: Grupo de nodos o servidores que agrupan un datacenter. Interconectan a través de una red. Aseguran alta disponibilidad.
- HA = Paralelización del trabajo + Replicación de los datos.

- Datacenter: Anillo. Colección de nodos, virtual o físico. Favorece la latencia. No tiene un SPOF.

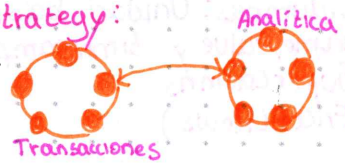


Datacenter

Tolerancia FR - 1

- Rack: Grupo de servidores
- Nodo: Componente básico donde se almacenan los datos
  - → MemTable: Especie de caché en memoria. Una vez lo escribe ahí, escribe en el:
  - → Commit log: Almacena la transacción para asegurar que esta se realizó (ej escritura)

  Una vez la MemTable se llena, se pasa por un proceso Flush y se guarda en almacenamiento en disco.
  - → SSTable: Almacena los datos escritos en disco. Se manejan versiones, no se sobre-escribe el dato.
- Protocolo comunicación inter-nodo (Gossip): Constantemente preguntar a sus nodos pares en qué estado están para verificar disponibilidad.
- Replica placement strategy: Define la estrategia a seguir para almacenar copias de los mismos datos en diferentes nodos. Accesibilidad y tolerancia a fallos.
  - → Simple Strategy
  - → Network Topology Strategy: Varios datacenters.



- Snitch: Define la topología que utilizan las estrategias de replicación para colocar las réplicas y dirigir las consultas de forma eficiente. A qué datacenter y rack pertenece cada nodo.

- **Partitioner:** Determina cómo se distribuyen los datos entre los nodos. Se debe replicar manteniendo un balanceo de carga.
  Algoritmo con una función de hash, con la PK asigna un valor qué determina en qué nodo se va a distribuir el dato.
  → Murmur3 Partitioner
  → Random Partitioner
  → Byte Ordered Partitioner

- **Cómo funciona la Replicación:** Nodo coordinador se encarga de llevar a cabo la replicación hasta asegurar el F.R configurado

- **Consistencia**
  → Cassandra no garantiza consistencia total de los datos. Es AP
  → Trade-OFF: Si quiero mayor consistencia voy a sacrificar rendimiento.
  (Pregunta a todos los nodos si tienen el mismo dato)

- **Niveles de consistencia:**
  → Any
  → One / Two / Three
  → QUORUM $quorum = \left(\frac{\#\,replic}{2}\right) + 1$
  → LOCAL_QUORUM (Datacenter local)
  → EACH_QUORUM (+ de 1 datacenter)
  → ALL (En todas las réplicas)

+ consistencia
+ rendimiento

MODELO DE DATOS

1. **Cluster:** Pueden contener múltiples keyspaces.
2. **Keyspace:** Esquema físico de B.D (relacionando con relacional) Agrupación de Column Families.
3. **Column Families:** Equivalente a tabla en modelo relacional. Cada entrada se identifica y accede mediante row-KEY.
4. **Columnas:** Unidad básica de almacenamiento. name, value y timestamp. Dato específico
5. **Supercolumns**
6. **Filas (Rows)**

- El timestamp es utilizado para la resolución de conflictos (Eventual Consistency).

- **Row-KEY:**
  → Partition key: Las filas con el mismo valor se almacenan en la misma partición del disco.
  → Clustering key: Determina el orden físico en el que se almacenan las filas.

ESPECIFICACIÓN CQL

- **CQL3:** Lenguaje de consulta y manipulación de datos de Cassandra. Intenta ser similar a SQL.
- Las Keyspaces son las bases de datos en Cassandra. Al crear se especifica
  1. Número de réplicas
  2. Estrategia de replicación

- **Tablas**
  1. Nombre de la tabla
  2. Nombre de las columnas
  3. Tipo de dato de las columnas
  4. PRIMARY KEY (partition key, clustering key)
     → Tanto la partition como la clustering key pueden ser compuestas.
     → Se puede prescindir de la clustering pero no de la partition key.

  1. Numéricos
  2. Texto
  3. Identificación
  4. Fecha
  5. .
  6. Otros tipos

- **Consultas**
  1. Proyección
  2. Tabla (solo 1, no joins)
  3. Criterios de búsqueda a cumplir: Solo utilizar columnas de partition key.
  4. Criterios de ordenamiento: Sobre columnas de clustering

```
SELECT idusuario
FROM usuarios
WHERE idusuario='1p67'

ORDER BY nombre
```

- **Insertados:** No se comprueba unicidad de la P.K. Se sobreescribe el dato (UPSERT)

- **Actualización, eliminación, índices**

MODELADO

- **Regla 1:** Distribuir los datos por todo el cluster → Equilibrio → Escoger la clave primaria adecuada.

- **Regla 2:** Minimizar el número de particiones a leer → Lectura más rápida.

- Una norma simple es crear una tabla para cada consulta.

- Se admite y recomienda la redundancia en beneficio del rendimiento en las consultas.