Análisis de Rendimiento de Operaciones CRUD en Cassandra (Simulado con DynamoDB local)

Contexto

Apache Cassandra es una base de datos NoSQL orientada a columnas, diseñada para manejar grandes cantidades de datos en entornos distribuidos. Su modelo de datos y arquitectura difieren considerablemente de sistemas relacionales, por lo que diseñar correctamente las claves de partición y evitar el uso indebido de índices secundarios resulta esencial para mantener un rendimiento óptimo.

En esta actividad se analizarán operaciones CRUD simuladas a través del servicio DynamoDB local utilizando Docker, dada su compatibilidad conceptual con Cassandra en términos de acceso distribuido y esquemas sin estructura estricta.

Entorno de Trabajo

Para esta simulación, se utilizó **DynamoDB local montado con Docker**, accesible a través del puerto `8000`. Esto permite realizar pruebas sin necesidad de acceder a una instancia real de AWS. Las operaciones se ejecutaron mediante la AWS CLI, con una tabla llamada `Comunas`.

Puerto de acceso: `http://localhost:8000`
Región simulada: `us-west-2`

Operaciones CRUD Evaluadas

A continuación, se presentan las operaciones simuladas y su análisis.

---

1. Inserción de una comuna

```
aws dynamodb put-item `

--table-name Comunas `

--item '{\"ComunaID\": {\"S\": \"002\"}, \"Nombre\": {\"S\":
\"La Florida\"}, \"Region\": {\"S\": \"Santiago\"}}' `

--endpoint-url http://localhost:8000 `

--region us-west-2
```

Análisis:

- Utiliza `ComunaID` como clave primaria, lo cual es adecuado si cada comuna es única.
- No se presentan problemas de rendimiento dado que el acceso se realiza por clave de partición directa.

Mejora sugerida:

- Validar que `ComunaID` sea efectivamente única y esté bien distribuida si se espera escalar a múltiples nodos.

---

2. Lectura de una comuna específica

```
aws dynamodb get-item `

--table-name Comunas `

--key '{\"ComunaID\": {\"S\": \"002\"}}' `

--endpoint-url http://localhost:8000 `

--region us-west-2
```

Análisis:

- Consulta óptima, ya que se accede directamente mediante la clave de partición.
- Sin implicaciones de rendimiento negativo.

Mejora sugerida:

- Ninguna. La operación es eficiente para lectura puntual.

---

3. Lectura de todas las comunas

```
aws dynamodb scan `
--table-name Comunas `
--endpoint-url http://localhost:8000 `
--region us-west-2
```

Análisis:

- El uso de `scan` es costoso en rendimiento y escalabilidad. Cassandra (y DynamoDB) no están optimizados para escaneos completos.
- No utiliza clave de partición, lo que obliga a recorrer todos los ítems.

Mejora sugerida:

- Crear una vista indexada si se necesitan agrupaciones frecuentes (por ejemplo, por `Region`).
- Evitar `scan` en producción; mejor usar filtros apoyados por claves.

---

4. Actualización de una comuna

```
aws dynamodb update-item `

--table-name Comunas `

--key '{\"ComunaID\": {\"S\": \"002\"}}' `

--update-expression "SET Nombre = :nuevoNombre, #R =
:nuevaRegion" `

--expression-attribute-names '{\"#R\": \"Region\"}' `

--expression-attribute-values '{\":nuevoNombre\": {\"S\": \"La
Florida Centro\"}, \":nuevaRegion\": {\"S\": \"Gran
Santiago\"}}' `

--endpoint-url http://localhost:8000 `

--region us-west-2
```

Análisis:

- Actualización eficiente porque se hace directamente por clave de partición.
- Correcta implementación de atributos con nombres reservados (`#R` para `Region`).

Mejora sugerida:

- Validar que los datos modificados sean consistentes con otras posibles tablas o vistas dependientes.

---

5. Eliminación de una comuna

```
aws dynamodb delete-item `

--table-name Comunas `

--key '{\"ComunaID\": {\"S\": \"002\"}}' `

--endpoint-url http://localhost:8000 `
```

```
--region us-west-2
```

Análisis:

- El borrado también es óptimo, ya que se hace por clave primaria.
- No afecta el rendimiento general.

Mejora sugerida:

- Implementar una estrategia de soft delete si se desea mantener trazabilidad histórica.

Conclusión

En este ejercicio se observaron las mejores prácticas en el diseño de operaciones CRUD para sistemas distribuidos como Cassandra, tomando como referencia una simulación en DynamoDB local. Se destaca que:

- El uso correcto de claves de partición es fundamental para asegurar rendimiento.
- Operaciones `get-item`, `put-item` y `delete-item` son eficientes si se usan adecuadamente.
- Las operaciones `scan` deben evitarse o reemplazarse por consultas más específicas.
- La arquitectura distribuida requiere un diseño anticipado de acceso a los datos.

Este tipo de análisis es crucial para asegurar la eficiencia y escalabilidad de aplicaciones en producción.