Managing Tombstones in Cassandra and Elastic Search

Mizbauddin Mohammad
Principal Software Engineer, Bentonville, AR, USA
mizba.md@gmail.com

Abstract-
This Article delves into advanced techniques for mitigating tombstones in Apache Cassandra when ingesting large datasets of retail supplier master data, enabling efficient search functionality through integration with Elasticsearch, this paper comprehensively explores the detrimental impact of tombstones on read performance and storage efficiency, and routine a multifaceted approach to minimize their creation and manage their lifecycle effectively.

Introduction

Managing large-scale, dynamic datasets in a retail environment presents unique challenges. Apache Cassandra, with its high availability and scalability, caters well to write-intensive workloads like ingesting and updating master data. However, its inherent design can lead to the accumulation of "tombstones" markers for deleted data. While tombstones play a crucial role in maintaining data consistency, their excessive presence can significantly hinder read performance and storage efficiency, especially when combine with elastic search for full-text search functionalities. This paper provides a comprehensive examination of advanced strategies for minimizing tombstones in Cassandra while optimizing it for retail master data and leveraging Elasticsearch for efficient search.

1. Understanding Tombstones and their Impact

Cassandra employs tombstones to track deleted data and ensure consistency across replicas. When data is deleted, a tombstone is inserted with the corresponding deletion timestamp, marking the data as unavailable for reads. While essential for historical accuracy, tombstones become problematic when they accumulate in large quantities. This is because the system needs to examine both live and tombstone data during read operations, leading to:

- **Performance degradation:** As the number of tombstones increases, read requests require scanning a larger volume of data, impacting query response times significantly.

- **Storage Inefficiency:** Tombstones occupy storage space, even though they represent non-existent data. This can lead to increased storage overhead and potentially higher operational costs.

2. Advanced Techniques for Minimizing Tombstones

2.1 Data modeling and Schema Design:

- **Partitioning Strategies:** Carefully select a partitioning strategy that aligns with your access patterns. Partitioning by frequently queried columns can minimize the number of tombstones scanned during reads.
- **Clustering Keys:** Design clustering keys to group frequently accessed fata together, reducing the need to scan through tombstones for unrelated data during reads.
- **Denormalization:** consider denormalizing data to reduce the need for joins and eliminate the creation of tombstones in the joined tables. However, this approach should be balanced with the potential drawbacks of increased data redundancy and consistency maintenance challenges.

2.2 Advanced Deletion and Update Strategies:

- **Tombstone Sweeping:** Utilize Cassandra's built-in tombstone sweeper tool to periodically remove expired tombstones, improving read performance and reclaiming storage space.
- **Counter Columns:** Employ counter columns for data that only needs to be incremented or decremented. This eliminates the need for deletes and prevents tombstone creation. However counter columns have limitations in terms of data types and querying capabilities.
- **Conditional Updates:** Utilize Conditional Updates (lightweight transactions) to perform updates only if certain conditions are met. This helps avoid unnecessary deletions and tombstones creation, especially when dealing with frequently updated data.

2.3 Compaction Strategies:

- **Leveled compaction Strategy:** This Strategy efficiently combines and discards tombstones during compactions, reducing tgeir overall footprint.
- **SizedTieredCompactionStrategy:** This strategy prioritizes compacting data from smaller, more frequently accessed sstables, which often contain higher concentration of tombstones.

- **Custom Compaction Logic:** Implement custom compaction logic to handle specifc data access patterns and tailor tombstone handling strategies accordingly. This approach requires advanced expertise and careful consideration to avoid unintended consequences.

3. Integration with Elasticsearch for Efficient Search:

Elasticsearch excels at full-text search and can be integrated with Cassandra to offload search functionalities. By indexing relevant data in Elasticsearch, users can perform efficient full-text queries without directly encountering tombstones in Cassandra, However, it is crucial to maintain data consistency between both systems to ensure accurate search results. Techniques like periodic bulk updates or near real-time data synchronization can be employed for this purpose.

4. Configuration:

- Interact indirectly with tombstone sweeper
    - **gc_grace_seconds** : This setting defines the grace period for tombstones. Once this time elapses, the tombstone is considered expired and eligible for removal during the sweeping process.
    - Compaction Strategy: Choosing the appropriate compaction strategyI (e.g. **LeveledCompactionStrategy**) influences how tombstones are handling during compaction cucles and ultimately removed.

5. Monitoring and Maintenance:

- **Cassandra Monitoring Service (CMS):** this tool offers various metrics related to tombstones, including their count, ratios and types. You can track them to understand effectiveness of your strategies.
- **sstablemetadata tool:** This tool allows you to analyze individual data structures(sstables) and view details like tombstone count and their impact on storage utilization.
- **Custom Monitoring tools:** You can develop custom tools or integrate existing monitoring solutions to track tombstone metrics alongside other system performance indicators.
- **Compact Scheduling:** Schedule Compactions at appropriate intervals to remove expired tombstones and optimize storage utilization.

- **Performance Analysis:** Analyze read and write performance metrics to assess the effectiveness of your tombstone management strategies and identify areas for further improvement.

Conclusion

By adopting a multifaceted approach that combines careful data modeling, advanced deletion and update strategies efficient compact strategies and thoughtful integration with Elasticsearch, Organizations can effectively manage tombstones in Cassandra while ingesting and managing large volumes of master data. This approach ensures optimal performance for both reads and writes, minimizes storage overhead, and facilitates efficient search functionalities with Elasticsearch.

Reference Articles

1. Managing Tombstones in Apache Cassandra: https://www.instaclustr.com/support/documentation/cassandra/cassandra-monitoring/tombstones-per-read/
2. Cassandra Query Language (CQL): SELECT statement: https://docs.datastax.com/en/drivers/java/2.0/com/datastax/driver/core/TokenRange.html (This page mentions tombstones in the context of filtering out deleted rows)
3. Cassandra, lists, and tombstones: https://jsravn.com/2015/05/13/cassandra-tombstones-collections/
4. Clarification about Cassandra tombstones and manual compaction: https://stackoverflow.com/questions/63660723/in-cassandra-are-partition- tombstones-inherently-less-expensive-compared-to-row
5. Cassandra: The Definitive Guide (Second Edition) by Eric A. Brewer (Chapter 12: Data Deletion and Tombstones)
6. Data Modeling for Cassandra by Sven Pedersen (Chapter 5: Tombstones and Garbage Collection)