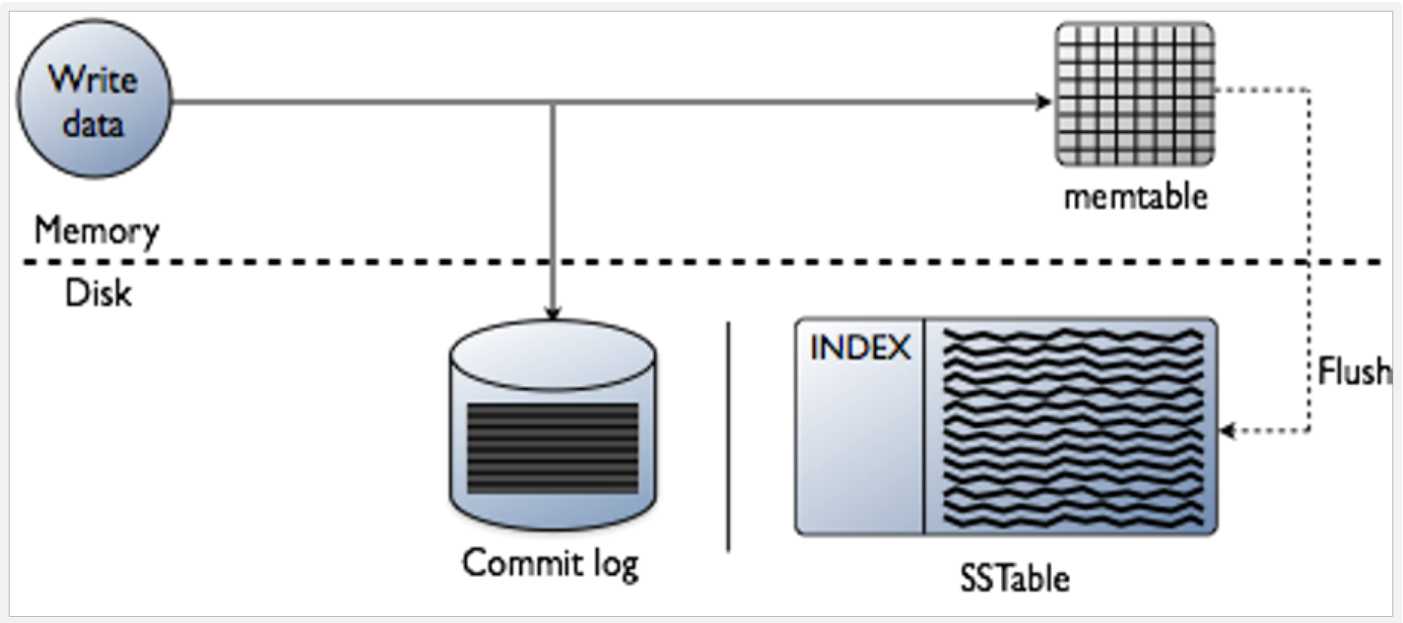
Quản lý và truy xuất dữ liệu trong Cassandra

1. Ghi dữ liệu trong Cassandra

Cassandra được tối ưu để việc ghi dữ liệu luôn có sự sẵn sàng cao và nhanh chóng. Trong khi đó, RDBMS có cấu trúc sao cho dữ liệu dư thừa là ít nhất, thông tin cần cho câu truy vấn được lưu trữ ở nhiều bảng có quan hệ với nhau. Chính vì lưu trữ dữ liệu như thế nên việc ghi dữ liệu tốn nhiều chi phí. Server database phải thực hiện thêm nhiều tác vụ để đảm bảo toàn vẹn dữ liệu thông qua nhiều bảng quan hệ. Vì vậy RDBMS thường có hiệu suất không cao khi ghi dữ liệu.



Luồng ghi dữ liệu trong Cassandra

Cassandra đầu tiên ghi dữ liệu vào commit log, sau đó là ghi vào cấu trúc bảng trong bộ nhớ gọi là memtable. Tác vụ ghi là thành công khi nó ghi vào commit log và vào memtable. Vì vậy có rất ít tác vụ tương tác đĩa tại thời điểm đọc ghi dữ liệu.

Việc ghi dữ liệu được thực hiện định kì vào disk trong một cấu trúc bảng nhất quán gọi là SSTable. Memtable và SSTable được tổ chức theo column family. Memtable được tổ chức bằng việc sắp xếp theo row key và được đẩy xuống SSTable một cách tuần tự.

SSTable là inmutable (không thay đổi), chúng không được ghi lại sau khi đã flush. Có nghĩa rằng một row được lưu trữ thông qua nhiều file trong SSTable. Tại thời điểm đọc, một row phải được kết hợp (combine) từ tất cả các SSTable trên đĩa để sinh ra dữ liệu được yêu cầu. Để tối ưu process

này, Cassandra đã dùng một cấu trúc trong bộ nhớ gọi là Bloom filter. Mỗi SSTable có một Bloom filter kết hợp với nó, dùng để kiểm tra nếu một row key được yêu cầu tồn tại trong SSTable trước khi làm tác vụ tìm kiếm trên đĩa.

2. Truy xuất dữ liệu đồng thời trong Cassandra

Không giống với RDBMS, Cassandra không hỗ trợ đầy đủ các tính năng trong ACID như không có khóa (locking) hay độc lập thực thi khi update nhiều dòng hoặc column family. ACID là từ viết tắt dùng để miêu tả 4 hành vi mà transaction phải đạt được trong RDBMS:

- Atomic: Đảm bảo giao dịch transaction đó phải thành công hoặc được quay lại (roll back) khi thất bại.
- Consistent: Đảm bảo transaction không thể để cơ sở dữ liệu ở trạng tháng thiếu nhất quán.
- Isolated: Transaction này phải độc lập với những transaction khác.
- Durable: Đảm bảo dữ liệu không bị mất mát khi có sự cố với hệ thống hoặc node server.

Cassandra đánh đổi giữa isolation và atomic để có khả năng đáp ứng cao trong việc thực hiện việc ghi dữ liệu nhanh. Trong Cassandra, việc ghi là một tác vụ atomic ở cấp độ dòng, nghĩa là update hay insert column của một row key được xem như là một tác vụ đọc. Cassandra không hỗ trợ transaction của một update trên nhiều dòng nên sẽ không thể quay lại (roll back) khi mà một tác vụ thành công trên một bảng sao nhưng lại thất bại trên một bản sao khác.

Cassandra sử dụng timestamp để cập nhật thời gian tác động gần nhất vào column. Timestamp được hỗ trợ bởi ứng dụng client. Timestamp cuối cùng luôn được chọn khi truy vấn/cập nhật data. Vì thế nếu nhiều giao dịch cùng update lên cũng một cột, update gần nhất sẽ được chọn.

Tác vụ ghi trong Cassandra là bền bỉ (durable). Tất cả tác vụ sẽ được ghi lại trong bộ nhớ và trong commit log trước khi chúng được biết như là một sự thành công. Nếu có thất bại (crash) hay server bị sự cố trước khi memtable được đẩy xuống đĩa. Commit log được dùng để phục hồi lại (recover) tác vụ ghi.

3. Insert và Update dữ liệu trong Cassandra

Nhiều cột có thể được insert tại cùng một thời điểm. Khi insert hay update column trong column family, ứng dụng đặc tả rowkey để nhận dạng cột nào được update. Rowkey tương tự như primary key, phải là duy nhất trong mỗi dòng của column family. Mặc dù vậy không giống như primary key, insert một duplicate row key sẽ không báo lỗi như primary key, nó đơn giản xem như là một tác vụ update.

Column được ghi đè nếu timestamp trong version mới của column gần hơn so với timestamp hiện tại. Vì thế timestamp chính xác là cần thiết nếu việc update xảy ra thường xuyên.

4. Delete dữ liệu trong Cassandra

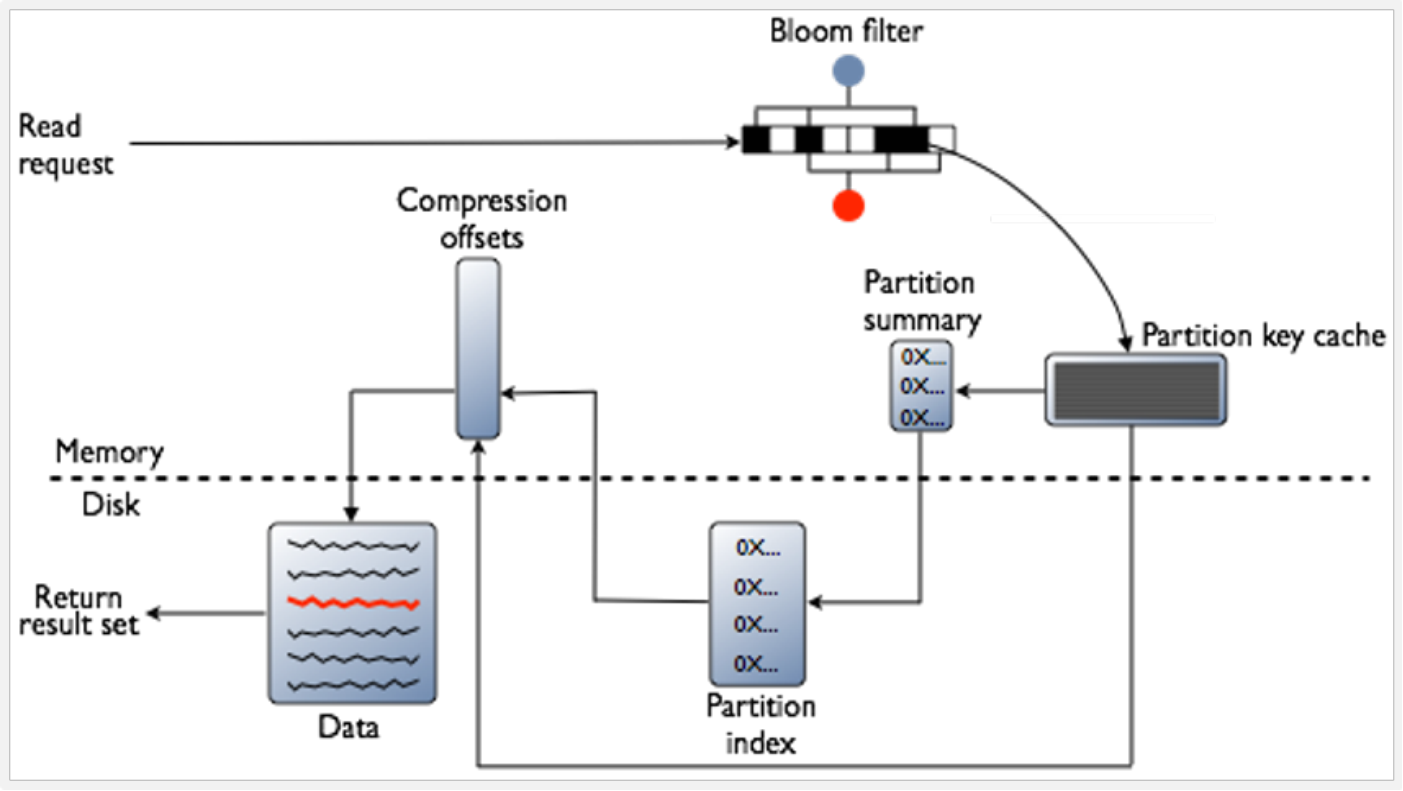
Khi thực hiện xóa (delete) dữ liệu trong Cassandra:

- Tác vụ xóa không thực hiện xóa dữ liệu trên đĩa ngay lập tức. Một lần SSTable được viết, nó sẽ không bị thay đổi (file không được update bởi DML), có nghĩa rằng cột bị xóa không bị xóa ngay lập tức. Hệ thống sẽ tạo một đánh dấu (maker) gọi là tombstone, để chỉ ra trạng thái của cột mới. Cột được đánh dấu bởi maker tồn tại trong khoảng thời gian được cấu hình trước, sau đó mới xóa vĩnh viễn bởi tác vụ compaction sau khi thời gian cấu hình bị quá hạn.
- Cột bị xóa có thể xuất hiện lại nếu routine node không chạy. Đánh dấu cột bị xóa bằng tombstone đảm bảo rằng một bản sao được đưa xuống tại thời điểm xóa sẽ bị xóa cuối cùng khi nó back up trở lại. Mặc dù vậy, nếu một node down lâu hơn thời điểm cấu hình để giữ tombstone thì node có thể mất dữ liệu bị xóa hoàn toàn, và nhân bản dữ liệu bị xóa 1 lần khi nó được back up trở lại. Để ngăn dữ liệu bị xóa xuất hiện lại, hệ thống cần phải điều chỉnh sửa chữa mỗi node.

Row key cho row bị xóa vẫn còn xuất hiện trong kết quả truy vấn (range query result). Khi bạn xóa một dòng trong Cassandra, nó đánh dấu tất cả các cột cho row key đó bằng tombstone. Cho đến khi nó bị xóa bởi compation bạn có một empty row key (1 row không có column). Những key bị xóa có thể được hiển thị trong get_range_slices(). Nếu ứng dụng client thực hiện câu truy vấn range lên row, ứng dụng có thể cũng cần phải loại bỏ đi những row không có cột nào.

5. Đọc dữ liệu trong Cassandra

Khi có một yêu cầu đọc lên row từ node, row phải được combine từ tất cả SSTable, cũng như là từ những memtable chưa flush xuống bộ nhớ để sinh ra dữ liệu yêu cầu. Để tối ưu quá trình này, Cassandra sử dụng cấu trúc gọi là Bloom filter cho mỗi SSTable, nó dùng để kiểm tra nếu dữ liệu cho những row được request tồn tại trong SSTable trước khi làm một số tác vụ truy vấn trên đĩa. Do đo, Cassandra có hiệu suất cao trong việc đọc khi so sánh với những hệ thống lưu trữ khác ngay cả khi có nhiều request đọc.



Luồng đọc dữ liệu trong Cassandra

Cũng giống như với một số cơ sở dữ liệu khác, tác vụ đọc là nhanh nhất khi dữ liệu yêu cầu nằm trong bộ nhớ memory. Mặc dù tất cả hệ thống lưu trữ mới thực hiện cache để cho phép truy xuất nhanh dữ liệu, nhưng không phải tất cả chúng đều được đảm bảo khi mà số tác vụ I/O vượt quá dung lượng của cache. Hiệu suất đọc dữ liệu của Cassandra cũng vậy, tuy nhiên, Cassandra dễ dàng khắc phục vấn đề này bằng cách thêm nhiều node cho cluster.

Revision #1
Created 26 June 2021 03:57:17 by Laptrinh.vn
Updated 26 June 2021 04:02:24 by Laptrinh.vn