Báo cáo kết quả tuần 1

Các chương 5,6,7 – Designing Data-Intensive Applications

Contents

[1. Chương 5 (Replication – Tái bản dữ liệu) 1](#_Toc200733853)

[1.1 Mục tiêu của replication 1](#_Toc200733854)

[1.2 Các mô hình replication 2](#_Toc200733855)

[1.2.1 Mô hình Single-leader Replication 2](#_Toc200733856)

[1.2.2 Mô hình Multi-leader Replication 2](#_Toc200733857)

[1.2.3 Mô hình Leaderless Replication 2](#_Toc200733858)

[1.3 Kết luân 2](#_Toc200733859)

[2. Chương 6 (Partitioning – Phân vùng dữ liệu) 3](#_Toc200733860)

[2.1 Mục tiêu của Partitioning 3](#_Toc200733861)

[2.2 Các kỹ thuật Partitioning 3](#_Toc200733862)

[2.3 Kết luận: 3](#_Toc200733863)

[3. Chương 7 (Transactions – Giao dịch) 3](#_Toc200733864)

[3.1 Khái niệm cơ bản về Transaction 3](#_Toc200733865)

[3.2 Concurrency Control – Kiểm soát đồng thời 3](#_Toc200733866)

[3.3 Distributed Transactions – Giao dịch phân tán 4](#_Toc200733867)

[3.4 Các hiện tượng bất thường trong isolation levels 4](#_Toc200733868)

[3.5 Kết luận 5](#_Toc200733869)

1. Chương 5 (Replication – Tái bản dữ liệu)
   1. Mục tiêu của replication

* Giữ dữ liệu gần hơn với người dùng (để giảm độ trễ)
* Cho phép hệ thống được hoạt động ngay khi có môt số bộ phận bị lỗi (để tăng tính khả dụng)
* Mở rộng số lượng máy có thể phục vụ các truy vẫn đọc(để tăng thông lượng đọc)
  1. Các mô hình replication
     1. Mô hình Single-leader Replication
* Tất cả yêu cầu đều được gửi đến leader, leader xử lý và ghi dữ liệu vào bộ nhớ của nó, khi đọc dữ liệu có thể đọc từ follower hoặc leader
* Ưu điểm : Dễ triển khai dễ kiểm soát,ghi dữ liệu một cách tuần tự hóa tại leader,
* Nhược điểm : Dễ quá tải cho leader nếu có nhiều bản ghi,leader bị lỗi thì cần thực hiện cơ chế bầu chọn leader (cần thời gian chờ),khi đọc bằng follower có thể ko có dữ liệu mới nhất
  + 1. Mô hình Multi-leader Replication
* Mỗi client gửi yêu cầu đến bất kỳ leader nào gần nó, mỗi leader ghi dữ liệu sau đó đồng bộ dữ liệu tới các leader khác (replicate), có thể dọc dữ liệu từ leader hoặc follower
* Ưu điểm : Hiệu suất khi ghi cao hơn (do có nhiều leader xử lý song song), Độ trễ thấp hơn(dữ liệu được ghi vào leader gần nhất), Tăng khả năng chịu lỗi (khi một leader chết thì sẽ có leader khác thay thế)
* Nhược điểm : Khó giữ nhất quán (vì có nhiều điểm ghi có thể xảy ra xung đột), Đồng bộ dữ liệu giữ các leader có thể gây trễ hoặc lỗi nếu mạng kém
  + 1. Mô hình Leaderless Replication
* Khi client muốn ghi dữ liệu nó sẽ gửi yêu cầu đến nhiều node cùng lúc(replica nodes), Ghi sẽ xem là thành công nếu ít nhất W node phàn hồi OK , Khi đọc dữ liệu client sẽ gửi yêu cầu tới R node, Nếu W + R > N sẽ đảm bảo tính nhất quán mạnh eventual consistency(N = tổng số bản sao, W = số node cần ghi thành công, R = số node cần đọc để trả dữ liệu)
* Ưu điểm : Tính sẵn sàng cao(không bị gián đoạn khi một node chết), Không có điểm nghẽn trung tâm, Hiệu suất cao nếu W và R hợp lý
* Nhược điểm: Khó đảm bảo tính nhất quán mạnh, Cần cơ chế xử lý các xung đột , Phức tạp trong quản lý và thiết kế hệ thống
  1. Kết luân
* Có khá nhiều thách thức trong việc duy trì sự nhất quán giữa các bản sao
* Cách giải quyết các xung đột phải tốt

1. Chương 6 (Partitioning – Phân vùng dữ liệu)
   1. Mục tiêu của Partitioning

* Giải quyết các vấn đề scaling khi dữ liệu lớn
* Cho phép hệ thống phân tán tải, tăng throughput
  1. Các kỹ thuật Partitioning
* Partitioning by Key Range: Chỉ định một phạm vi khóa liên tục(từ một giá trị tối thiểu đến một giá trị tối đa) cho mỗi phân vùng
* Partitioning by Hash of Key: dùng hàm băm để phân vùng dữ liệu
* Partitioning and Secondary Indexes:Có 2 cách là **Partitioning Secondary Indexes by Document**, **Partitioning Secondary Indexes by Term**
  1. Kết luận:

Có nhiều thách thức trong Partitioning:

* **Rebalancing** khi thêm các node mới
* **Skew**( khi dữ liệu ko đồng đều sẽ gây ra bottleneck)
* Partitioning ảnh hưởng đến joins, transactions, đòi hỏi thiết kế 1 cách cẩn thận

1. Chương 7 (Transactions – Giao dịch)
   1. Khái niệm cơ bản về Transaction

* Giao dịch là để đảm bảo tính toàn vẹn dữ liệu trong môi trường phần tán khi các các phần mềm hoặc phần cứng cơ sở dữ liệu có thể bị lỗi
* Các thuộc tính ACID: **Automicity**: Đảm bảo tất cả hoặc không là gì cả (Nếu xảy ra lỗi giữa chừng một chuỗi các lần ghi, giao dịch sẽ bị hủy bỏ và các lần ghi được thực hiện cho đến thời điểm đó sẽ bị loại bỏ), **Consistency**: Giữ trạng thái luôn hợp lệ , **Isolation**: Không bị can thiệp giữa các giao dịch, **Durability**: Đảm bảo dữ liệu sẽ được lưu trữ sau khi commit
  1. Concurrency Control – Kiểm soát đồng thời
* Hệ thống cần cho phép nhiều người dùng làm việc cùng lúc, nhưng vẫn phải đảm bảo dữ liệu chính xác.
* Các kỹ thuật Concurrency Control : **Lock-based**: Khóa dữ liệu để ngăn xung dột (dễ bị tắc nghẽn), **Optimistic concurrency control** : không khóa nhưng kiểm tra xung đột khi commit, **Snapshot Isolationg**: Mỗi giao dịch “nhìn thấy” bản snapshot riêng của dữ liệu(nhanh nhưng có thể bị “write skew”)
  1. Distributed Transactions – Giao dịch phân tán
* Khi dữ liệu nằm ở nhiều node (database phân tán), thì giao dịch càng phức tạp
* Vấn đề đặt ra : Cần đảm bảo tất cả các node đồng ý commit hoặc rollback, Nếu 1 node thất bại thì các hệ thống có thể bị block
* Cơ chế phổ biến là Two-phase commit (2PC) **: phase 1**: các node vote “ready”, **phase 2**: nếu tất cả đồng ý thì commit, nếu tất cẩ từ chối thì rollback
  1. Các hiện tượng bất thường trong isolation levels
* **Dirty Read**: Giao dịch A đọc dữ liệu mà giao dịch B đã ghi nhưng chưa commit (nguy cơ dữ liệu không nhất quán, khó lần vết lỗi có thể tránh bằng Read committed trở lên )
* **Dirty Write**: Hai giao dịch ghi lên cùng một dữ liệu cùng lúc (nguy cơ ghi đè lẫn nhau gây mất dữ liệu có thể giải quyết bằng các hệ thống cần locking hoặc serialization)
* **Read skew**: Một Một giao dịch đọc hai giá trị liên quan, nhưng mỗi giá trị ở một thời điểm khác nhau gây mâu thuẫn logic ( nguy cơ vi phạm ràng buộc logic trong ứng dụng có thể tránh bằng cách dùng Snapshot Isolation trờ lên)
* **Write skew** : Hai giao dịch đọc cùng dữ liệu, rồi ghi dữ liệu mới dựa trên giả định cũ dẫn đến vi phạm ràng buộc logic (nguy cơ vi phạm tính toàn vẹn dữ liệu có thể tránh bằng cách dùng Serializable isolation ở mức cao nhất)
* **Lost Update**: Hai giao dịch cùng đọc một giá trị, sau đó cùng ghi lại dựa trên bản sao cũ, dẫn đến mất một bản cập nhật (không khóa ghi dẫn mất dữ liệu người dùng có thể dùng locking , serializable isolation đề phòng ngừa)
* **Phantom Read** : Một giao dịch đọc tập hợp dữ liệu thỏa điều kiện nào đó, rồi sau đó cùng giao dịch đó đọc lại và thấy nhiều kết quả hơn hoặc ít hơn do giao dịch khác vừa thêm/xóa( Gây **lỗi logic** nếu ứng dụng giả định tập dữ liệu không thay đổ có thể dùng Serializable isolation hoặc một số DB hỗ trợ để giải quyết)
  1. Kết luận
* Cho thấy vai trò thiết yếu của giao dịch trong việc đảm bảo tính toàn vẹn và nhất quán của dữ liệu trong các hệ thống dữ liệu hiện đại
* Bằng cách phân tích các hiện tượng như dirty read, lost update, write skew và phantom read cho thấy tại sao việc kiểm soát đồng thời (concurrency control) là rất khó khăn nhưng cần thiết.