**Cơ chế xử lý song song trong Cơ sở dữ liệu phân tán**

Trong **CSDL phân tán**, xử lý song song là một yếu tố quan trọng giúp cải thiện **hiệu suất, khả năng mở rộng và độ tin cậy**. Hệ thống có thể xử lý nhiều truy vấn hoặc giao dịch đồng thời bằng cách phân chia công việc trên nhiều nút trong cụm (cluster).

Dưới đây là các cơ chế xử lý song song trong CSDL phân tán:

**1. Mô hình xử lý song song trong CSDL phân tán**

**1.1. Shared-Nothing Architecture (Kiến trúc không chia sẻ)**

* Mỗi nút (node) có **CPU, bộ nhớ, và ổ đĩa riêng**.
* Dữ liệu được **phân mảnh** (sharding) trên các nút để xử lý song song.
* Ví dụ: **Google Spanner, Apache Cassandra, CockroachDB**.

**1.2. Shared-Disk Architecture (Kiến trúc chia sẻ đĩa)**

* Các nút chia sẻ **cùng một kho dữ liệu** nhưng có bộ nhớ và CPU riêng.
* Truy vấn có thể chạy song song mà không cần di chuyển dữ liệu.
* Ví dụ: **Oracle RAC (Real Application Clusters)**.

**1.3. Shared-Memory Architecture (Kiến trúc chia sẻ bộ nhớ)**

* Các nút **chia sẻ cùng bộ nhớ RAM**, giúp giảm trễ khi truy vấn dữ liệu chung.
* Phù hợp với hệ thống có **tài nguyên giới hạn nhưng yêu cầu tốc độ cao**.
* Ví dụ: **IBM DB2 Parallel Edition**.

**2. Các kỹ thuật xử lý song song trong CSDL phân tán**

**2.1. Parallel Query Processing (Xử lý truy vấn song song)**

* Truy vấn SQL có thể **phân tán trên nhiều nút** để tăng tốc độ xử lý.
* **Chiến lược phân mảnh dữ liệu** ảnh hưởng lớn đến hiệu suất xử lý truy vấn song song.

📌 **Ví dụ:**

sql

Sao chépChỉnh sửa

SELECT COUNT(\*)

FROM orders

WHERE amount > 1000;

* Nếu dữ liệu được **phân mảnh ngang**, truy vấn này có thể chạy đồng thời trên nhiều nút và tổng hợp kết quả.
* Nếu dữ liệu được **phân mảnh dọc**, có thể cần nhiều bước ghép dữ liệu, làm giảm hiệu suất.

**2.2. Parallel Transaction Processing (Xử lý giao dịch song song)**

* Hệ thống có thể **xử lý nhiều giao dịch đồng thời** mà không làm ảnh hưởng đến tính nhất quán.
* Các kỹ thuật đồng bộ phổ biến:
  + **Two-Phase Locking (2PL)** để tránh xung đột ghi.
  + **Multi-Version Concurrency Control (MVCC)** để đọc không khóa.
  + **Two-Phase Commit (2PC)** để đảm bảo tính toàn vẹn dữ liệu.

📌 **Ví dụ:** Nếu hai người dùng cập nhật cùng một tài khoản ngân hàng từ hai chi nhánh khác nhau, hệ thống phải đảm bảo rằng cả hai giao dịch đều được xử lý đúng thứ tự.

**2.3. Data Replication & Load Balancing (Nhân bản và cân bằng tải dữ liệu)**

* **Replication** giúp dữ liệu có sẵn trên nhiều nút để xử lý song song.
* **Load Balancing** giúp phân phối truy vấn đến nút có ít tải nhất.
* Các mô hình replication phổ biến:
  + **Leader-Follower Replication** (Dữ liệu ghi vào Leader, đọc từ Follower).
  + **Multi-Leader Replication** (Nhiều nút có thể ghi dữ liệu).
  + **Quorum-based Replication** (Cần đạt số lượng nút tối thiểu để ghi dữ liệu).

📌 **Ví dụ:** Trong hệ thống phân tán như **Apache Cassandra**, **DynamoDB**, dữ liệu được nhân bản trên nhiều cụm để tăng tốc độ xử lý song song.

**2.4. Distributed Joins (Xử lý phép JOIN trong hệ thống phân tán)**

* Khi dữ liệu được **phân mảnh trên nhiều nút**, JOIN có thể gặp vấn đề về hiệu suất.
* Các chiến lược JOIN phổ biến:
  + **Broadcast Join**: Sao chép bảng nhỏ đến tất cả các nút.
  + **Shuffle Join**: Chia lại dữ liệu để JOIN được thực hiện cục bộ trên mỗi nút.
  + **Replicated Join**: Sao chép toàn bộ bảng đến các nút.

📌 **Ví dụ:**

sql

Sao chépChỉnh sửa

SELECT users.name, orders.amount

FROM users

JOIN orders ON users.id = orders.user\_id;

* Nếu users và orders nằm trên các nút khác nhau, hệ thống cần chuyển dữ liệu giữa các nút để thực hiện JOIN.

**3. Thách thức trong xử lý song song**

🚧 **1. Đồng bộ dữ liệu giữa các nút**: Khi nhiều giao dịch chạy song song, cần đảm bảo tính nhất quán.  
🚧 **2. Phân vùng mạng (Network Partitioning)**: Một số nút có thể không kết nối