**BÁO CÁO: HỆ THỐNG PHÂN TÁN VỚI SLEEKDB**

**I. GIỚI THIỆU**

**1.1. Tổng quan về hệ thống phân tán**

Hệ thống phân tán là một tập hợp các máy tính kết nối với nhau qua mạng, hoạt động như một hệ thống duy nhất. Các hệ thống này có thể chia sẻ tài nguyên, xử lý dữ liệu song song, và cung cấp độ tin cậy cao thông qua cơ chế sao lưu dữ liệu. Mô hình này giúp nâng cao hiệu suất, giảm tải cho từng máy chủ và đảm bảo tính sẵn sàng của hệ thống.

**1.2. Giới thiệu về SleekDB**

SleekDB là một cơ sở dữ liệu NoSQL được xây dựng trên nền tảng PHP, không yêu cầu máy chủ riêng như MySQL hay MongoDB. SleekDB sử dụng hệ thống lưu trữ tệp tin JSON, giúp dễ dàng triển khai và mở rộng. Với cơ chế lưu trữ linh hoạt, SleekDB có thể hoạt động hiệu quả trong các hệ thống nhỏ đến vừa mà không cần đến một hệ quản trị cơ sở dữ liệu phức tạp.

**1.3. Mục tiêu nghiên cứu**

Báo cáo này tập trung vào việc triển khai và mở rộng SleekDB để hỗ trợ hai tính năng quan trọng trong hệ thống phân tán:

* **Multiple Node Replication**: Nhân bản dữ liệu trên nhiều máy chủ để đảm bảo tính an toàn và sẵn sàng của dữ liệu.
* **Sharding**: Chia nhỏ và phân phối dữ liệu trên nhiều node để tối ưu hóa hiệu suất truy vấn và cân bằng tải.

**II. TÍNH NĂNG MULTIPLE NODE REPLICATION**

**2.1. Định nghĩa**

Multiple Node Replication là kỹ thuật giúp sao chép dữ liệu từ một node chính đến nhiều node khác nhằm đảm bảo tính sẵn sàng và độ tin cậy của hệ thống. Khi một node gặp sự cố, dữ liệu vẫn có thể được truy xuất từ các node replica, giúp hệ thống duy trì hoạt động mà không bị gián đoạn.

**2.2. Cách triển khai trong SleekDB**

* Khi một bản ghi được thêm vào node chính, nó sẽ tự động gửi bản sao đến các node replica.
* Sử dụng cURL để gửi dữ liệu qua HTTP POST đến các node replica.
* Các node replica nhận dữ liệu, lưu trữ vào hệ thống của chúng và đảm bảo dữ liệu luôn đồng bộ.
* Khi truy vấn, nếu node chính bị lỗi, dữ liệu vẫn có thể được truy vấn từ các node replica.

**2.3. Cấu trúc mã nguồn**

private function replicateData(array $data) {

foreach ($this->replicaNodes as $node) {

$url = $node . "/replicate.php";

$this->sendHttpRequest($url, $data);

}

}

**2.4. Kiểm thử**

* **Bước 1**: Thêm dữ liệu vào node chính bằng setup.php.
* **Bước 2**: Kiểm tra thư mục /data/ của các node replica để xác nhận dữ liệu đã được sao chép.
* **Bước 3**: Dừng node chính (8001) và thử truy vấn dữ liệu từ các node replica để kiểm tra tính toàn vẹn.

**III. TÍNH NĂNG SHARDING**

**3.1. Định nghĩa**

Sharding là kỹ thuật giúp chia nhỏ dữ liệu và phân phối chúng trên nhiều node khác nhau để tối ưu hóa hiệu suất truy vấn. Thay vì lưu trữ tất cả dữ liệu trên một node duy nhất, hệ thống sẽ xác định node lưu trữ dựa trên một khóa xác định (shard key).

**3.2. Cách triển khai trong SleekDB**

* Khi dữ liệu mới được thêm vào, hệ thống sử dụng thuật toán băm (crc32) để xác định node lưu trữ.
* Mỗi node lưu trữ một phần dữ liệu, giúp tối ưu hóa truy vấn và giảm tải cho từng máy chủ.
* Khi truy vấn dữ liệu, hệ thống chỉ cần truy vấn node lưu trữ dữ liệu thay vì toàn bộ hệ thống.

**3.3. Cấu trúc mã nguồn**

public function getShardNode($shardKey) {

$index = crc32($shardKey) % count($this->shardNodes);

return $this->shardNodes[$index];

}

**3.4. Kiểm thử**

* **Bước 1**: Chèn dữ liệu với nhiều shardKey khác nhau và xác định node lưu trữ bằng test\_sharding.php.
* **Bước 2**: Kiểm tra thư mục /data/ trên từng node để xác nhận dữ liệu được phân phối đúng.
* **Bước 3**: Gửi truy vấn đến từng node và kiểm tra phản hồi để đảm bảo hệ thống hoạt động chính xác.

**IV. ĐÁNH GIÁ HỆ THỐNG**

**4.1. Ưu điểm**

* **Tăng độ tin cậy**: Replication giúp bảo vệ dữ liệu ngay cả khi một node bị lỗi.
* **Cải thiện hiệu suất**: Sharding giúp phân tán tải giữa các node, giảm áp lực cho từng máy chủ.
* **Dễ mở rộng**: Hệ thống có thể thêm nhiều node mới mà không cần thay đổi kiến trúc cốt lõi.

**4.2. Hạn chế**

* **Cần tối ưu truyền dữ liệu**: Việc sử dụng cURL có thể gây độ trễ nếu số lượng node replica lớn.
* **Quản lý sharding phức tạp hơn**: Khi số node tăng lên, việc cân bằng tải và di chuyển dữ liệu giữa các node cần được tối ưu.

**V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

**5.1. Kết luận**

Hệ thống đã được mở rộng thành công với hai tính năng quan trọng: Replication và Sharding. Điều này giúp SleekDB có thể hoạt động hiệu quả hơn trong môi trường phân tán, nâng cao tính sẵn sàng của dữ liệu và tối ưu hóa hiệu suất truy vấn.

**5.2. Hướng phát triển**

* **Tích hợp Load Balancer** để tối ưu hóa truy vấn dữ liệu và cân bằng tải giữa các node.
* **Viết API RESTful** để hỗ trợ truy xuất dữ liệu dễ dàng hơn.
* **Hỗ trợ caching** để tăng tốc độ truy vấn và giảm tải cho các node.