CS112

Phạm Ngọc Phú Thịnh – 24521703

Phạm Nguyễn Duy Anh – 24520004

Câu 1. Tìm hiểu về load balancing. Load Balancer hoạt động như thế nào ? Tìm hiểu và trình bày cơ chế hoạt động của các thuật toán Load Balancer (Round Robin và Least Connection). Phân tích và nhận xét 2 thuật toán đó.

Câu 2. Trình bày vấn đề mà Raft giải quyết trong hệ thống phân tán, mô tả cách hoạt động của Raft. Trong một hệ thống, khi nào nên sử dụng Raft ?

Câu 1. Tìm hiểu về Load Balancing

1. Khái niệm Load Balancing (Cân bằng tải)

Load Balancing là kỹ thuật phân phối đồng đều lưu lượng truy cập (request) đến nhiều máy chủ (server) trong một hệ thống, nhằm giảm tải cho từng server riêng lẻ, tăng hiệu suất xử lý, đảm bảo tính ổn định và sẵn sàng cao (high availability), giúp hệ thống hoạt động liên tục ngay cả khi một server gặp sự cố.

Ví dụ: Một website có 5 server cùng chạy ứng dụng. Khi người dùng truy cập, Load Balancer sẽ quyết định chuyển request đó đến server nào, sao cho hệ thống làm việc hiệu quả nhất.

2. Cách Load Balancer hoạt động

Load Balancer nằm ở giữa client và server, có thể hiểu là một “người điều phối”. Khi người dùng gửi yêu cầu đến hệ thống:

1. Request được gửi tới Load Balancer.

2. Load Balancer chọn một server phù hợp (dựa theo thuật toán).

3. Request được chuyển tiếp đến server đó để xử lý.

4. Server xử lý xong → gửi kết quả lại cho Load Balancer → trả về client.

Sơ đồ minh họa:

Client → Load Balancer → Server 1
→ Server 2
→ Server 3

3. Các thuật toán Load Balancer phổ biến

(a) Round Robin

Cơ chế hoạt động: Load Balancer phân phối request theo vòng tuần tự:
Request 1 → Server 1
Request 2 → Server 2
Request 3 → Server 3
Request 4 → quay lại Server 1.

Ưu điểm:

- Đơn giản, dễ cài đặt.
- Hiệu quả khi các server có cấu hình tương đương và request có tải tương tự nhau.

Nhược điểm:

- Không xét đến tải thực tế trên mỗi server.
- Có thể gây quá tải nếu một server xử lý chậm.

(b) Least Connection

Cơ chế hoạt động: Load Balancer chọn server có ít kết nối đang hoạt động nhất. Mỗi khi có request mới, nó được chuyển đến server rảnh nhất.

Ví dụ:
Server 1: 5 connections
Server 2: 3 connections
Server 3: 2 connections → Request mới → Server 3

Ưu điểm:

- Phân phối tải thực tế hơn Round Robin.
- Hiệu quả trong hệ thống có request nặng nhẹ khác nhau.

Nhược điểm:

- Cần theo dõi số kết nối thường xuyên, tốn tài nguyên hơn.
- Có thể lệch nhẹ nếu thời gian xử lý khác nhau.

4. Phân tích & Nhận xét

Tiêu chí	Round Robin	Least Connection
Nguyên lý	Phân phối tuần tự	Phân phối dựa vào số kết nối đang hoạt động

Công bằng về số lượng request	Có	Có (nhưng thông minh hơn)
Công bằng về tải thực tế	Không	Có
Độ phức tạp	Thấp	Cao hơn
Hiệu quả trong hệ thống không đồng nhất	Kém	Tốt
Ứng dụng thực tế	Hệ thống nhỏ, tải đều	Hệ thống lớn, tải không đồng đều

5. Kết luận

• Round Robin phù hợp với hệ thống đơn giản, tải ổn định, các server tương đương nhau.
• Least Connection phù hợp với hệ thống thực tế, nơi mức độ xử lý của request khác nhau và server có cấu hình không đồng đều.
• Trong thực tế, các Load Balancer hiện đại như Nginx, HAProxy, AWS ELB thường kết hợp hoặc tùy chỉnh linh hoạt giữa các thuật toán này.

Câu 2. Raft – Vấn đề giải quyết và cách hoạt động

1. Vấn đề mà Raft giải quyết trong hệ thống phân tán:

Trong hệ thống phân tán, dữ liệu được lưu trữ trên nhiều nút (node). Khi một hoặc vài nút gặp sự cố (crash, mất kết nối…), hệ thống cần đảm bảo rằng:
- Tất cả các nút còn lại vẫn nhất quán (consistency).
- Hệ thống vẫn có thể tiếp tục hoạt động (availability).

Đây chính là bài toán “đồng thuận phân tán” (Distributed Consensus). Thuật toán Raft được thiết kế để đạt được sự đồng thuận giữa các nút trong hệ thống phân tán, đảm bảo tất cả các nút đều thống nhất cùng một trạng thái dữ liệu, ngay cả khi có lỗi xảy ra.

2. Cách hoạt động của Raft:

Raft chia hệ thống thành 3 vai trò chính:
- Leader (chủ): nhận yêu cầu từ client, cập nhật log, và gửi log đó đến các Follower.
- Follower (theo): chỉ nhận lệnh từ Leader, không tự hành động.
- Candidate: tạm thời trở thành ứng viên để bầu làm Leader khi Leader hiện tại bị lỗi.

Quy trình hoạt động chính của Raft gồm 3 giai đoạn:

a. Leader Election (Bầu chọn Leader):
- Khi hệ thống khởi động, tất cả nút đều là Follower.
- Nếu một Follower không nhận được tín hiệu “heartbeat” từ Leader trong một thời

gian nhất định, nó sẽ trở thành Candidate và bắt đầu bầu cử.
- Các nút khác bỏ phiếu, và nút nhận được quá nửa số phiếu sẽ trở thành Leader mới.

b. Log Replication (Nhân bản nhật ký):
- Leader nhận lệnh từ client, ghi vào log của mình và gửi bản sao log đó đến các Follower.
- Khi đa số Follower xác nhận đã ghi log, Leader coi lệnh đó là hợp lệ và áp dụng vào trạng thái hệ thống.

c. Safety (Đảm bảo an toàn):
- Raft đảm bảo rằng không có hai Leader khác nhau cùng tồn tại trong một nhiệm kỳ.
- Mọi quyết định được thực hiện đều nhất quán giữa các nút hợp lệ.

3. Khi nào nên sử dụng Raft:

Raft thường được sử dụng trong các hệ thống phân tán cần đảm bảo tính nhất quán dữ liệu cao, ví dụ:
- Hệ thống lưu trữ phân tán (etcd, Consul, TiKV, CockroachDB...)
- Dịch vụ cấu hình tập trung.
- Các hệ thống cần đồng thuận mạnh và chịu lỗi tốt.