**BÁO CÁO**

1. **Bài toán đồng thời cao**

- Bài toán cao tải: cao tải không lường trước và cao tải có lường trước, ở đây xét chủ yếu là bài toán cao tải có lường trước

- concurrency và parallel:

+, concurrency: khả năng quản lý nhiều tác vụ, thực thi các nhiệm vụ xen kẽ

+, parallelism: thực hiện nhiều nhiệm vụ, các nhiệm vụ chạy cùng lúc, thực sự

- **Ví dụ về bài toán đồng thời: 3 request cùng yêu cầu 1 slot duy nhất, các request gần như liên tục chỉ các nhau 0.001s, chỉ 1 request lấy được slot đó**

⇒ Có 2 hướng giải quyết chính hay dùng đó là sử dụng Queue và Redis

**a, Queue**

- Client gửi request ⇒ backend controller ⇒ queue ⇒ được đọc và xử lý rồi lưu xuống DB

⇒ Tuy nhiên dùng queue thì nó phản hồi chậm do rất nhiều request khác cũng được lưu trong queue

⇒ Truy cập trực tiếp vào DB dễ dẫn đến nghẽn cổ chai

**b, Sử dụng Redis**

- Redis(Remote dictionary server) là 1 hệ thống lưu trữ dữ liệu dạng key-value hoạt động trong bộ nhớ, là 1 công nghệ cụ thể(phần mềm, database,...)

- Cache là nơi lưu trữ tạm dữ liệu thường xuyên đc truy cập để tăng tốc độ response và giảm tải cho hệ thống gốc, cache có thể nằm trong Ram, Redis,...

⇒ **Cache là 1 cơ chế dùng để tăng tốc dữ liệu. Redis là 1 công cụ có thể được dùng để triển khai cache và làm nhiều thứ khác**

- Trước khi hệ thống khởi động, dữ liệu slot từ DB sẽ được load vào Redis trước, giúp truy vấn nhanh tránh mỗi lần phải vào lại DB để lấy lưu vào Redis

- Cache ghi trước, Db ghi sau

- Flow: Client ⇒ gửi request ⇒ Background Job ghi DB ⇒ tiến hành đồng bộ cache và DB

+, **B1**: client gửi request lấy slot, backend nhận yêu cầu nhưng không ghi DB ngay mà sử dụng Redis để kiểm tra

⇒ sử dụng khoá phân tán Distributed Lock, nó có 2 cơ chế là Acquire Lock và Unlock

- Distributed Lock lưu tên tài nguyên cần khoá(ở đây là id), value(id của client/process đang giữ lock), TTL(time to live: thời gian được giữ khoá để tránh tình trạng bế tắc - deadlock)

+,**B2:** Request đầu tiên tới sẽ được giữ Lock và set lại vào redis để khoá lock. Các Request tiếp theo đến thấy Lock đang bị giữ rồi sẽ phải chờ đến khi nào nó được unlock thì mới có thể lấy đc Lock

⇒ giúp cho 3 thằng thì chỉ có 1 thằng được giữ Lock

+,**B3**: Ngay sau khi cache thành công thì backend response ngay cho client mà chưa cần đợi DB xử lý ⇒ cải thiện UX

+,**B4**: Sau đó các request được đẩy vào Queue, các request này được worker(tiến trình chuyên đọc job từ quêu và thực thi chúng) đọc và xử lý, kiểm tra lại redis, tiến hành lưu trữ thật xuống DB

⇒ Với cách sử dụng Redis này thì:

+, Hệ thống chịu được nhiều request cùng 1 lúc

+, Tránh được bế tắc-deadlock

+, Người dùng không phải chờ DB xử lý

- Cơ chế ACK( Acknowledgement): giúp xác nhận đã nhận hoặc xử lý đã thành công

+, Khi worker đọc 1 job từ queue, hệ thống sẽ gửi ACK cho queue sau khi job xử lý xong, nếu không gửi queue hiểu là job chưa xong/worker crash và có thể đưa job cho worker khác xử lý ⇒ đảm bảo tính tin cậy cho hệ thống

- Mô hình Meta Data:

+, trong hệ thống phân tán: là dữ liệu mô tả cấu trúc và trạng thái các node(master, slave, sentinel)

+, trong hệ thống thông tin nói chung: là dữ liệu mô tả về dữ liệu

khác(các bảng, cột, kiểu dữ liêu,..)

- Mô hình Redis Master-Slave Replication: là mô hình sao chép dữ liệu trong đó có 1 Redis master và nhiều Redis slave, những Redis slave sẵn sàng lên thay Redis master nếu nó bị hỏng

+, Khi client write dữ liệu thì master sẽ thực hiện lệnh đó

+, Khi client read dữ liệu thì slave sẽ thực hiện

+, Slave nhận và đồng bộ dữ liệu từ master

⇒ Giảm tải áp lực

- Rate Limiter: Cơ chế giới hạn số lần 1 hành động được thực hiện trong 1 khoảng thời gian nhất định

+, Rate Limiter ID: khoá định danh dùng để xác định đối tượng bị giới hạn ⇒ Redis sẽ lưu bộ đếm hoăc token bucket cho mỗi ID riêng ( id, ip, api key,...)

* Object storage: 1 kho lưu trữ chứa dữ liệu phi cấu trúc dạng object thay vì file hoặc block, truy cập qua api, được tối ưu cho cloud

+, 1 Object bao gồm 3 phần chính:

* Object Data: Dữ liệu gốc(ảnh, file, log, backup,...)
* Metadata: Thông tin mô tả về Object(loại file, ng tạo, thgian, quyền truy cập,...)
* Object Id(key): Định danh duy nhất để truy cập Object đó

1. **Bài số 2**

* Session management: là cơ chế giúp quản lý phiên làm việc, cơ chế giúp duy trì trạng thái người dùng giữa các request http, do http là giao thức không nhớ trạng thái

+, Quy trình hoạt động cơ bản

* User đăng nhập ⇒ gửi username + password
* Server xác thực thành công ⇒ tạo session id là mã của phiên làm việc
* Server lưu thông tin về session vào DB/Redis
* Server gửi session id về cho client và lưu ở trong cookie
* Mỗi request sau đó, client gửi lại session id ⇒ server nhận diện

- Fingerprint: Device fingerprint là 1 chuỗi mã duy nhất được sinh ra từ thông tin đặc trưng của thiết bị và trình duyệt mà người dùng đang sử dụng. Nó giúp server nhận diện 2 lần đăng nhập có đến từ 1 máy hay không mà không cần dùng cookie hay ip

+, Fingerprint được tạo thành bằng cách kết hợp nhiều thông tin có sẵn như trình duyệt, loại thiết bị, hệ thống, ip của mạng, …. Tất cả thông tin này được hash lại thành 1 mã ngắn. Tuy nhiên nếu 2 máy có cấu hình y hệt nhau thì sẽ tạo ra 1 fingerprint gần như trùng nhau

⇒ **Cách gây khó chịu nhất cho 2 user share chung 1 tài khoản**

* Khi client đăng nhập sẽ gửi thông tin đăng nhập + finger print
* Server xác thực và tạo sessionid lưu vào redis
* Khi có đăng nhập mới, server kiểm trta userif này đã có session chưa. Nếu có rồi và fingerprint khác ⇒ Xoá session cũ ⇒ Tự động logout người trước sau đó tạo 1 session mới có sesion id và fingerprint mới