1. **Làm thế nào để đảm bảo tính nhất quán dữ liệu trong hệ thống Microservices?**

Để đảm bảo tính nhất quán dữ liệu trong hệ thống Microservices, tôi thường áp dụng các phương pháp sau:

1. **Saga Pattern**: Sử dụng Saga để quản lý các giao dịch phân tán. Mỗi service thực hiện một phần của giao dịch và gửi event để service tiếp theo xử lý. Nếu có lỗi, các compensating transaction sẽ được thực hiện để rollback.
2. **Eventual Consistency**: Chấp nhận tính nhất quán cuối cùng bằng cách sử dụng message queue (như RabbitMQ hoặc Kafka) để đồng bộ dữ liệu giữa các service.
3. **Distributed Locking**: Sử dụng distributed lock (ví dụ: Redis Lock) để đảm bảo chỉ một service có thể thay đổi dữ liệu tại một thời điểm.
4. **Database per Service**: Mỗi service có database riêng để tránh tranh chấp dữ liệu.
5. **Bạn hiểu gì về kiến trúc event-driven? Làm thế nào để xử lý message loss trong RabbitMQ?**

Kiến trúc event-driven là một mô hình trong đó các service giao tiếp thông qua events (sự kiện). Khi một service thực hiện một hành động, nó sẽ publish một event, và các service khác subscribe để xử lý event đó.

Để xử lý message loss trong RabbitMQ, tôi thường áp dụng các phương pháp sau:

**+ Message Acknowledgment**: Đảm bảo consumer gửi acknowledgment sau khi xử lý message thành công.

**+ Persistent Messages**: Đánh dấu message là persistent để RabbitMQ lưu message vào disk, tránh mất dữ liệu khi server crash.

**+ Dead Letter Exchange (DLX):** Sử dụng DLX để xử lý các message không thể xử lý được sau một số lần retry.

**+ Monitoring và Alerts**: Sử dụng công cụ như Prometheus và Grafana để theo dõi hệ thống và cảnh báo sớm khi có sự cố.

1. **Làm thế nào để thiết kế một API RESTful tốt?**

**RESTful Principles**: Sử dụng đúng HTTP methods (GET, POST, PUT, DELETE) và status codes (200, 201, 400, 404, 500).

**Versioning**: Sử dụng versioning (ví dụ: /api/v1/resource) để đảm bảo tương thích ngược.

**Documentation**: Sử dụng Swagger để tạo tài liệu API chi tiết và dễ hiểu.

**Security**: Áp dụng OAuth2 hoặc JWT để bảo mật API.

**Pagination và Filtering**: Hỗ trợ pagination và filtering để tối ưu hiệu suất.

Thiết kế một hệ thống có thể chịu được **150 triệu request/ngày** (khoảng **1,736 request/giây**) đòi hỏi phải áp dụng các nguyên tắc kiến trúc hiệu suất cao, khả năng mở rộng (scalability), và độ tin cậy (reliability).

**1. Đảm bảo perfomance**

**Cân bằng tải với Load Balancer**: Load Balancer là một thiết bị (phần cứng hoặc phần mềm) cho phép cân bằng tải đến nhiều server.

A diagram of a network

AI-generated content may be incorrect.

**Phân tán dữ liệu với Content Delivery Network (CDN):** CDN là một mạng lưới các server được phân bố trên nhiều khu vực.

**Caching**: Cache là một kĩ thuật để tăng tốc độ đọc dữ liệu, bằng cách lưu dữ liệu sẵn vào cache server để tăng tốc độ đọc ở những lần sau.

**2. Đảm bảo Availability**

**Master/Slave:** Thay vì chỉ chạy 1 server, ta chạy 2 hoặc nhiều hơn. 1 server chính gọi là master, các server còn lại là slave. Khi master có vấn đề (sập nguồn hay crash), một slave sẽ được chỉ định để lên thay thế master.

**Replication**: Thường được kết hợp chung với Load Balancer. Code của ứng dụng sẽ được deploy lên nhiều server. Khi có 1 server die, load balancer sẽ chuyển request sang server khác, đảm bảo request vẫn được thực hiện.

**3. Đảm bảo scalability**

**Vertical Scaling**: Tăng sức mạnh phần cứng cho server

**Horizontal Scaling**: Thay vì tăng sức mạnh cho 1 server, ta thêm nhiều server vào hệ thống và chạy cùng lúc.

**1. Câu hỏi về Spring Framework**

**Spring Core**

1. **Spring là gì? Nêu các tính năng chính của Spring Framework.**
2. **IoC (Inversion of Control) và DI (Dependency Injection) là gì? Cho ví dụ.**
3. **Bean trong Spring là gì? Làm thế nào để định nghĩa một bean?**
4. **Sự khác biệt giữa @Component, @Service, @Repository, và @Controller?**
5. **Spring ApplicationContext là gì? Nó khác gì với BeanFactory?**
6. **Các scope của bean trong Spring? Giải thích từng scope.**
7. **Làm thế nào để cấu hình Spring mà không sử dụng XML?**
8. **Spring Profiles là gì? Làm thế nào để sử dụng chúng?**
9. **AOP (Aspect-Oriented Programming) trong Spring là gì? Cho ví dụ về cách sử dụng @Aspect.**
10. **Sự khác biệt giữa @Autowired và @Resource?**

**Spring Boot**

1. Spring Boot là gì? Tại sao nó được sử dụng phổ biến?
2. Cách tạo một ứng dụng Spring Boot đơn giản?
3. Spring Boot Auto-Configuration là gì? Nó hoạt động như thế nào?
4. Làm thế nào để cấu hình port và context path trong Spring Boot?
5. Spring Boot Actuator là gì? Nó dùng để làm gì?
6. Cách sử dụng @SpringBootApplication annotation?
7. Sự khác biệt giữa @RestController và @Controller?
8. Làm thế nào để tích hợp database (ví dụ: MySQL) vào Spring Boot?
9. Cách xử lý exception trong Spring Boot?
10. Làm thế nào để triển khai ứng dụng Spring Boot lên cloud (ví dụ: AWS)?

**Spring Data JPA**

1. JPA là gì? Nó khác gì với Hibernate?
2. Cách sử dụng @Entity, @Table, @Id, và @GeneratedValue trong Spring Data JPA?
3. Sự khác biệt giữa CrudRepository và JpaRepository?
4. Làm thế nào để viết custom query trong Spring Data JPA?
5. Cách xử lý transaction trong Spring Data JPA?

**Spring Security**

1. Spring Security là gì? Nó dùng để làm gì?
2. Cách cấu hình Spring Security để bảo vệ ứng dụng?
3. Sự khác biệt giữa Authentication và Authorization?
4. Cách sử dụng JWT (JSON Web Token) trong Spring Security?
5. Làm thế nào để triển khai OAuth2 trong Spring Security?

**2. Câu hỏi về Spring Integration**

1. Spring Integration là gì? Nó dùng để làm gì?
2. Các thành phần chính trong Spring Integration (Message, Channel, Endpoint)?
3. Làm thế nào để tích hợp Spring Integration với hệ thống bên ngoài (ví dụ: JMS, FTP)?
4. Sự khác biệt giữa Point-to-Point Channel và Publish-Subscribe Channel?
5. Cách sử dụng @Transformer, @Filter, và @Router trong Spring Integration?
6. Làm thế nào để xử lý lỗi trong Spring Integration?
7. Cách tích hợp Spring Integration với Kafka hoặc RabbitMQ?
8. Spring Integration DSL là gì? Cho ví dụ về cách sử dụng.
9. Làm thế nào để monitor và debug các luồng tích hợp trong Spring Integration?
10. Cách sử dụng Spring Integration để xử lý bất đồng bộ?

**3. Câu hỏi về Java Core**

**Java Basics**

1. Sự khác biệt giữa JDK, JRE, và JVM?
2. Các tính năng chính của Java?
3. Sự khác biệt giữa == và .equals()?
4. String trong Java là immutable. Tại sao?
5. Sự khác biệt giữa String, StringBuilder, và StringBuffer?
6. Các kiểu dữ liệu nguyên thủy (primitive types) trong Java?
7. Wrapper classes trong Java là gì? Tại sao chúng được sử dụng?
8. Sự khác biệt giữa final, finally, và finalize?
9. Cách sử dụng try-with-resources trong Java?
10. Sự khác biệt giữa ArrayList và LinkedList?

**OOP (Object-Oriented Programming)**

1. 4 nguyên lý cơ bản của OOP? Giải thích từng nguyên lý.
2. Sự khác biệt giữa lớp trừu tượng (abstract class) và interface?
3. Overloading và Overriding là gì? Cho ví dụ.
4. Sự khác biệt giữa public, private, protected, và default access modifiers?
5. Cách sử dụng super và this trong Java?
6. Sự khác biệt giữa composition và inheritance?
7. Polymorphism là gì? Cho ví dụ.
8. Cách sử dụng static trong Java?
9. Sự khác biệt giữa instanceof và getClass()?
10. Cách sử dụng enum trong Java?

**Collections**

1. Các interface chính trong Java Collections Framework?
2. Sự khác biệt giữa List, Set, và Map?
3. Cách sử dụng HashMap? Làm thế nào nó hoạt động?
4. Sự khác biệt giữa HashMap và Hashtable?
5. Cách sử dụng Comparator và Comparable?
6. Sự khác biệt giữa ArrayList và Vector?
7. Cách sử dụng Iterator và ListIterator?
8. Concurrent Collections trong Java là gì? Cho ví dụ.
9. Cách sử dụng Stream API trong Java 8?
10. Sự khác biệt giữa parallelStream và stream?

**Multithreading**

1. Cách tạo một thread trong Java?
2. Sự khác biệt giữa Thread và Runnable?
3. Cách sử dụng synchronized trong Java?
4. Sự khác biệt giữa wait(), notify(), và notifyAll()?
5. Cách sử dụng ExecutorService để quản lý thread pool?
6. Sự khác biệt giữa Callable và Runnable?
7. Cách sử dụng Future và CompletableFuture?
8. Deadlock là gì? Làm thế nào để tránh deadlock?
9. Cách sử dụng volatile trong Java?
10. Sự khác biệt giữa ReentrantLock và synchronized?

**Exception Handling**

1. Sự khác biệt giữa checked và unchecked exceptions?
2. Cách sử dụng try-catch-finally?
3. Sự khác biệt giữa throw và throws?
4. Cách tạo một custom exception?
5. Cách sử dụng try-with-resources?

**4. Câu hỏi kỹ năng mềm**

1. Bạn đã từng làm việc trong một dự án lớn với nhiều thành viên chưa? Bạn đã đóng góp như thế nào?
2. Làm thế nào để bạn quản lý thời gian và ưu tiên công việc khi có nhiều task cùng lúc?
3. Bạn đã từng gặp xung đột trong team chưa? Bạn đã giải quyết như thế nào?
4. Bạn đã từng xử lý sự cố trên production chưa? Hãy kể về một tình huống cụ thể.
5. Làm thế nào để bạn học một công nghệ mới?

**1. Câu hỏi nâng cao về Spring Framework**

**Spring Core**

1. Làm thế nào để tạo một custom BeanPostProcessor trong Spring? Cho ví dụ.
2. Cách sử dụng @Conditional annotation trong Spring? Khi nào nên sử dụng nó?
3. Spring FactoryBean là gì? Khi nào nên sử dụng nó?
4. Cách sử dụng @Import và @ImportResource trong Spring?
5. Làm thế nào để tạo một custom scope trong Spring? Cho ví dụ.

**Spring Boot**

1. Cách tạo một custom Spring Boot Starter?
2. Làm thế nào để tối ưu hóa hiệu suất của ứng dụng Spring Boot?
3. Cách sử dụng @ConfigurationProperties để binding dữ liệu từ file cấu hình?
4. Làm thế nào để tích hợp Spring Boot với GraphQL?
5. Cách sử dụng Spring Boot Actuator để monitor và quản lý ứng dụng?

**Spring Security**

1. Cách triển khai OAuth2 Resource Server trong Spring Security?
2. Làm thế nào để tích hợp Spring Security với Keycloak?
3. Cách sử dụng @PreAuthorize và @PostAuthorize để kiểm soát quyền truy cập?
4. Làm thế nào để triển khai Multi-Tenancy Security trong Spring Security?
5. Cách sử dụng Spring Security để bảo vệ WebSocket?

**Spring Data JPA**

1. Cách sử dụng @EntityGraph để tối ưu hóa truy vấn JPA?
2. Làm thế nào để sử dụng @Query với native query trong Spring Data JPA?
3. Cách triển khai Auditing (ghi lại thay đổi) trong Spring Data JPA?
4. Làm thế nào để sử dụng @Modifying và @Transactional trong Spring Data JPA?
5. Cách sử dụng Specification và Criteria API trong Spring Data JPA?

**2. Câu hỏi nâng cao về Spring Integration**

1. Làm thế nào để xử lý lỗi (error handling) trong Spring Integration?
2. Cách sử dụng Aggregator và Splitter trong Spring Integration?
3. Làm thế nào để tích hợp Spring Integration với Apache Kafka?
4. Cách sử dụng MessageChannel và MessageHandler để xử lý message?
5. Làm thế nào để triển khai một hệ thống event-driven sử dụng Spring Integration?

**3. Câu hỏi nâng cao về Java Core**

**Java Memory Model**

1. Java Memory Model (JMM) là gì? Giải thích về happens-before relationship.
2. Cách sử dụng volatile để đảm bảo tính nhất quán trong đa luồng?
3. Sự khác biệt giữa synchronized và ReentrantLock? Khi nào nên sử dụng từng loại?
4. Làm thế nào để tránh deadlock trong Java?
5. Cách sử dụng ThreadLocal trong Java? Khi nào nên sử dụng nó?

**Concurrency**

1. Cách sử dụng CompletableFuture để xử lý bất đồng bộ trong Java?
2. Làm thế nào để sử dụng ForkJoinPool và RecursiveTask?
3. Cách sử dụng StampedLock để tối ưu hóa hiệu suất trong đa luồng?
4. Làm thế nào để triển khai một custom ThreadPoolExecutor?
5. Cách sử dụng Phaser để đồng bộ hóa các luồng trong Java?

**Performance Tuning**

1. Làm thế nào để tối ưu hóa hiệu suất của ứng dụng Java?
2. Cách sử dụng Java Flight Recorder (JFR) để phân tích hiệu suất?
3. Làm thế nào để sử dụng Garbage Collection tuning để cải thiện hiệu suất?
4. Cách sử dụng JMH (Java Microbenchmark Harness) để đo lường hiệu suất?
5. Làm thế nào để sử dụng Profiling Tools như VisualVM hoặc YourKit?

**4. Câu hỏi nâng cao về Java Collection Framework**

1. Cách sử dụng ConcurrentHashMap để đảm bảo thread-safe?
2. Làm thế nào để triển khai một custom Comparator và Comparable?
3. Cách sử dụng Stream API để xử lý dữ liệu trong Java 8?
4. Làm thế nào để sử dụng Collectors để nhóm dữ liệu trong Stream API?
5. Cách sử dụng Spliterator để chia nhỏ và xử lý dữ liệu song song?

**5. Câu hỏi nâng cao về Hibernate**

Hibernate Core

1. Cách sử dụng @BatchSize để tối ưu hóa truy vấn trong Hibernate?
2. Làm thế nào để sử dụng @Fetch và @FetchMode để điều khiển fetching strategy?
3. Cách sử dụng @Filter để lọc dữ liệu trong Hibernate?
4. Làm thế nào để sử dụng @NamedQuery và @NamedNativeQuery trong Hibernate?
5. Cách sử dụng @Version để triển khai optimistic locking trong Hibernate?

**Performance Tuning**

1. Làm thế nào để tối ưu hóa hiệu suất của ứng dụng sử dụng Hibernate?
2. Cách sử dụng Second-Level Cache trong Hibernate?
3. Làm thế nào để sử dụng Query Cache để tối ưu hóa truy vấn?
4. Cách sử dụng Batch Processing để cải thiện hiệu suất khi xử lý dữ liệu lớn?
5. Làm thế nào để sử dụng Stateless Session trong Hibernate?

**Advanced Mapping**

1. Cách sử dụng @Inheritance để triển khai inheritance mapping trong Hibernate?
2. Làm thế nào để sử dụng @ElementCollection để mapping collections trong Hibernate?
3. Cách sử dụng @JoinTable và @JoinColumn để mapping quan hệ nhiều-nhiều?
4. Làm thế nào để sử dụng @Embeddable và @Embedded để mapping composite keys?
5. Cách sử dụng @Formula để tính toán giá trị động trong Hibernate?

**6. Câu hỏi về Design Patterns và Best Practices**

1. Làm thế nào để áp dụng Dependency Injection (DI) trong Spring một cách hiệu quả?
2. Cách sử dụng Factory Pattern để tạo đối tượng trong Spring?
3. Làm thế nào để áp dụng Proxy Pattern trong Spring AOP?
4. Cách sử dụng Builder Pattern để tạo đối tượng phức tạp trong Java?
5. Làm thế nào để áp dụng Observer Pattern trong Spring Integration?

**7. Câu hỏi về Testing**

1. Cách sử dụng Mockito để viết unit test trong Spring?
2. Làm thế nào để sử dụng Spring Boot Test để viết integration test?
3. Cách sử dụng Testcontainers để kiểm thử ứng dụng với database thật?
4. Làm thế nào để sử dụng JUnit 5 để viết unit test trong Java?
5. Cách sử dụng WireMock để mock các dịch vụ bên ngoài trong Spring Boot?