Ajit Gupta Follow 2 min read · Jun 23, 2025

🚀 Interview Question : Java & Spring Boot Insights🌟
✅ Q. “How Would You Handle 1M Requests Per Minute?”

Handling 1 million requests per minute (~16,666 requests/second) requires a scalable, resilient, and high-performance system design.

Here's a high-level system design + concurrency strategies in Java to handle.

✅ High-Level System Design

🏗️ 1. Architecture Overview
Client -> API Gateway -> Load Balancer -> Web/Application Servers -> Service Layer -> DB / Cache / MQ

Horizontally scalable (containers, Kubernetes)
REST APIs using Spring Boot, Netty, or Vert.x

4. Asynchronous Processing (for high throughput)
Use Message Queues like Kafka, RabbitMQ
Queue writes, slow operations

5. Database
Use read-replicas for scale
NoSQL (e.g., Cassandra, MongoDB) for write-heavy
SQL (e.g., PostgreSQL, MySQL) with caching

6. Caching Layer
Redis / Memcached
Cache hot data, responses, rate limits

7. CDN
Offload static content (images, videos, JS) to a CDN (Cloudflare, Akamai)

🚀 Java-Based Concurrency Tips

1. Use Thread Pooling Efficiently
Avoid creating threads manually. Use:
ExecutorService executor = Executors.newFixedThreadPool(100);
✅ For reactive workloads, consider ForkJoinPool or CompletableFuture.

2. Use Asynchronous & Non-Blocking I/O
Use Spring WebFlux, Project Reactor, or Vert.x for reactive programming.
Prefer non-blocking I/O over traditional servlet thread-per-request.

3. Avoid Blocking Calls
Avoid using Thread.sleep, join, or any long CPU-bound operations in the request thread.
Use CompletableFuture.supplyAsync() for non-blocking service calls.

4. Leverage Caching
Use @Cacheable (Spring) or Redis for frequent, expensive queries:
@Cacheable("userProfile")
public UserProfile getUser(String userId) {
// DB call
}

5. Connection Pooling
Use HikariCP (default in Spring Boot) for efficient DB connections:
spring.datasource.hikari.maximum-pool-size=50

6. Use Backpressure / Throttling
Protect downstream systems with:
Bulkhead pattern (limit concurrent calls)
Circuit breaker (e.g., Resilience4j)
Rate limit using token bucket or leaky bucket algorithm

7. Monitoring & Auto-scaling
Use Prometheus + Grafana
Autoscale containers/pods based on CPU/memory/load

🔄 Sample Load Handling Strategy (for 1M RPM)

Tier : Scale Required
API Gateway : Auto-scaled, throttling
App Servers : 10-20 instances with 100-200 threads each
DB : Read replicas, connection pool, cache
Cache : Redis Cluster for horizontal scaling
MQ : Kafka for async buffering (e.g.,emails,logs)

Written by Ajit Gupta

165 followers · 208 following

Follow

Responses (2)

Write a response

What are your thoughts?

Rajaganesh
Jun 26

Thanks .can you share the same for 1M request per second.

Reply

Paileemallesh
Jun 25

Thank you

Reply

More from Ajit Gupta

Ajit Gupta

🚀 Java Microservices Interview Cheat Sheet (20 Q&A)

Jun 16 46

Ajit Gupta

All Oracle Senior Java Developer Interview Questions

Jun 13 11

Ajit Gupta

☕ Java & Spring Interview Questions for Experienced...

🚀 Microservices with Spring Boot—Step-by-Step Beginner Guide

1d ago 16

Ajit Gupta

Java + Spring Boot Interview Questions Real Experience

Jul 4 5

See all from Ajit Gupta

Recommended from Medium

In Stackademic by Kavya's Programming Path

I Reviewed 500 Pull Requests—Here's What Every Java Dev Gets...

From Optional disasters to stream overkill, these are the Java sins haunting your code...

6d ago 131 6

In Java Programming by Pudari Madhavi

12 Multithreading Java Interview Questions

Break down complex Java multithreading questions into simple, interview-ready...

Jun 2 111 2

In Coding Odyssey by Shivam Srivastava

Qualcomm Java Interview Experience—2

Interview of Professional with 6+ Years of Experience

5d ago 13

Full Stack With Ram

Top 20 Java 8 Streams Coding Interview Questions Every...

Java 8 brought a paradigm shift to how we write code, and one of its most powerful...

Jul 3 1

In ILLUMINATION by Jaytech

Ajit Gupta

Java Developer Interview

Scaling Spring Boot to 1 Million Requests Per Second: A Battle-...

handling massive traffic is not just about writing clean code -it's about....

Jun 4 156 4

Hey everyone!

May 20

See more recommendations