面试官最爱问：你线上 QPS 是多少？你怎么知道的？

原创 江小北 程序员江小北 2025年03月20日 08:30

说在前面

在小北的读者群里，无数小伙伴为了拿高薪，疯狂卷大厂面试、卷架构师面经。有一个问题，几乎成了面试必考题：

> **“你们系统QPS多少？你怎么知道的？如果每天几千万请求，系统怎么部署？”**

这题堪称面试界的“钉子户”，**10个面试9个问**！
今天，小北就用**多年开发经验**，带大家手撕这个题，从理论到实操，从监控到部署，**让你面试时怼得面试官心服口服**！

> **文末附有满分面试回答**

一、QPS扫盲

1.1 什么是QPS？

QPS = 每秒查询数（Query Per Second），简单说就是**服务器1秒能接多少活**。
举个例子：

- 小摊煎饼：老板1分钟能做10个煎饼 → **QPS≈0.17**（10/60）
- 肯德基窗口：1分钟能出餐50份 → **QPS≈0.83**

QPS越高，说明系统越能扛！

1.2 QPS、TPS、RT的关系

🔷 TPS（每秒事务数）

- 事务 = 一套完整操作（比如下单：选商品→支付→扣库存）
- **TPS = 1秒能完成多少套完整操作**

🔷 RT（响应时间）

- 从你点外卖到拿到餐的时间 → **RT=30分钟**
- 系统处理一个请求的时间 → **RT=200ms**

三者的数学关系

```
QPS = 并发数 / RT
TPS = QPS × 每个请求的事务数
```

举个栗子：

- 系统RT=50ms，100人同时点外卖 → QPS = 100 / 0.05 = 2000
- 每个下单请求包含3个事务（扣库存、支付、发短信） → TPS = 2000 × 3 = 6000

1.3 必须知道的性能公式

📈 二八定律算峰值QPS

假设系统每天1000万请求：

```
峰值QPS = (总请求 × 80%) / (24h × 20% × 3600秒)
= (10,000,000 × 0.8) / (6 × 3600)
≈ 370 QPS
```

这意味着：系统必须能稳定扛住370 QPS，否则高峰期必崩！

二、实战！监控QPS的4步大法

2.1 第一步：给Spring Boot装“监控探头”

加依赖

在项目的 `pom.xml` 里塞两个“监控神器”：

```
<!-- 健康检查 -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<!-- Prometheus数据采集 -->
```

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

改配置

`application.yml` 里开放监控端口：

```
management:
  endpoints:
    web:
      exposure:
        include: "*"  # 开放所有监控指标
  metrics:
    web:
      server:
        request:
          metric-name: http.server.requests # 统一指标名
```

启动项目，查看指标

访问 `http://你的IP:端口/actuator/prometheus` ，你会看到：

```
http_server_requests_seconds_count{uri="/api/order",status="200"} 358
http_server_requests_seconds_sum{uri="/api/order",status="200"} 28.5
```

这表示：

- `/api/order` 接口被调用了358次（ `count` ）
- 总耗时28.5秒（ `sum` ）
- 平均RT = 28.5s / 358 ≈ 80ms

2.2 第二步：部署Prometheus（监控数据中心）

安装Prometheus

去官网下个压缩包，解压后改 `prometheus.yml` ：

```
scrape_configs:
  - job_name: '你的服务'
    metrics_path: '/actuator/prometheus'
    static_configs:
      - targets: ['你的IP:端口'] # 改成你的服务地址
```

启动命令

```
./prometheus --config.file=prometheus.yml
```

访问 `http://localhost:9090` 就能看到监控数据啦！

2.3 第三步：用Grafana搞个炫酷看板

配置数据源

1. 安装Grafana后，进入页面
2. 左侧菜单 → 齿轮图标 → Data Sources → 选Prometheus
3. 填地址 `http://localhost:9090` → Save & Test

创建QPS监控图

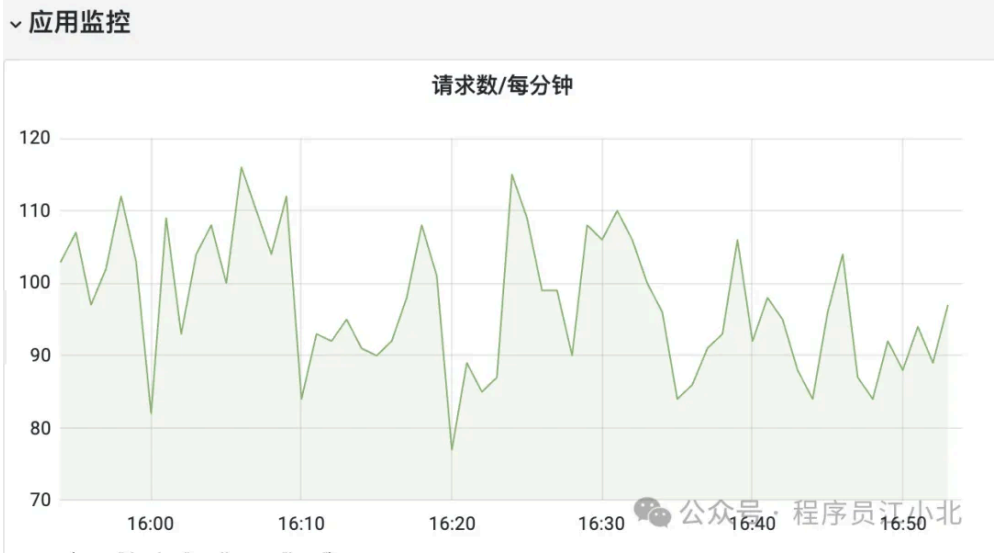
1. 新建Dashboard → Add panel
2. 输入PromQL：

```
rate(http_server_requests_seconds_count{uri="/api/order"}[5m])
```

这个公式的意思是：**统计5分钟内，/api/order接口每秒的请求数**

1. 设置图表名称 → Apply

效果图：



QPS监控曲线示例

2.4 第四步：高级操作——区分不同接口

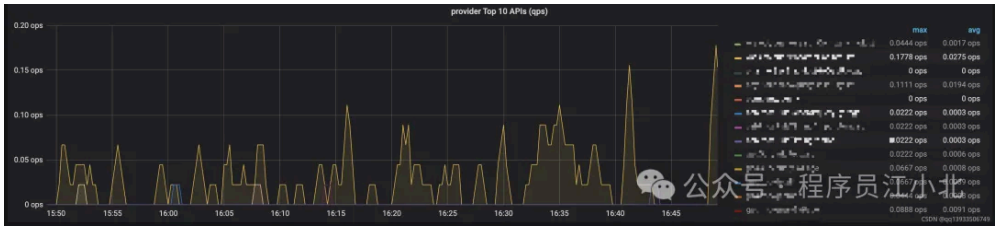
如果系统有多个接口（比如订单、支付、用户），可以在Grafana里：

1. 使用 `legend` 字段显示 `uri` 标签
2. 添加多个PromQL，按接口拆分

```
rate(http_server_requests_seconds_count{uri=~"/api/.*"}[5m])
```

最终效果：

- 清晰看到哪个接口最忙
- 快速定位性能瓶颈



三、千万级流量的部署秘籍

3.1 初级方案：单体架构升级

纵向扩容（Scale Up）

- 升级服务器：CPU从4核→16核，内存8G→64G
- 适用场景：初期流量小（日活<10万）

缺点：

- 单点故障 → 服务器一挂，全线崩溃
- 成本高 → 顶级配置的服务器价格指数级上涨

3.2 进阶方案：微服务拆分

拆分策略

1. **按业务拆分**：订单服务、用户服务、支付服务
2. **读写分离**：订单读库、订单写库
3. **热点隔离**：把秒杀功能独立成单独服务

技术选型

- 服务注册：Nacos、Consul
- 服务通信：OpenFeign、gRPC
- 负载均衡：Ribbon、Spring Cloud LoadBalancer

3.3 高级方案：弹性伸缩

K8s + 云服务

1. 用Kubernetes部署服务
2. 设置HPA（自动扩缩容）：

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: order-service
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 80
```

效果： CPU超过80%时，自动新增Pod，最多10个实例

3.4 全链路优化

分层优化策略

层级	优化手段	效果
网关层	Nginx限流 + 缓存静态资源	减少50%后端请求
服务层	线程池隔离 + 熔断降级	防止雪崩，提升可用性
缓存层	Redis集群 + 热点数据本地缓存	降低DB压力90%
DB层	分库分表 + 读写分离	提升查询速度300%

四、面试满分指南

4.1 满分回答模板

面试官： 你们系统QPS多少？
你：

> 我们的订单系统峰值QPS在1200左右，通过Prometheus+Grafana实时监控。
> 为了支撑每天2000万请求，做了三方面优化：
>
> 1. **服务层**：订单服务拆分为8个Pod，HPA根据CPU自动扩缩
> 2. **缓存层**：Redis集群缓存热点订单，命中率85%
> 3. **DB层**：MySQL分16个库，通过ShardingSphere路由
>
> 这是我们的监控看板截图，可以看到.....（拿出手机展示）

面试官OS： 这哥们是真干过！

4.2 防坑指南

千万别说

- “QPS大概几千吧，没具体测过” → 不专业
- “直接加服务器搞定” → 缺乏技术深度

加分话术

- “我们通过全链路压测确定了系统瓶颈”
- “用Sentinel做了慢调用熔断，防止级联故障”

五、总结

要想QPS答得好，三板斧不能少：

1. **监控**：Prometheus+Granfana实时追踪
2. **优化**：缓存、分库、限流组合拳
3. **部署**：K8s弹性伸缩应对流量高峰

小北私藏精品　热门推荐

小北联合公司合伙人，一线大厂在职架构师耗时9个月联合打造了**《2024年Java高级架构师课程》**本课程对标外面3万左右的架构培训课程，分10个阶段，目前已经更新了**181G**视频，已经更新**1000+**个小时视频，**一次购买，持续更新，无需2次付费**

近期技术热文

count(*)、count(1)哪个更快？面试必问：通宵整理的十道经典MySQL必问面试题

腾讯三面：40亿个QQ号，如何用1GB内存处理？

面试官问：MySQL自增ID用完了，怎么办？

用雪花算法生成订单 ID，现在我有点后悔了

第3版：互联网大厂面试题

包括 Java 集合、JVM、多线程、并发编程、设计模式、算法调优、Spring全家桶、Java、MyBatis、ZooKeeper、Dubbo、Elasticsearch、Memcached、MongoDB、Redis、MySQL、RabbitMQ、Kafka、Linux、Netty、Tomcat、Python、HTML、CSS、Vue、React、JavaScript、Android 大数据、阿里巴巴等大厂面试题等、等技术栈！

阅读原文： 高清 7701页大厂面试题 PDF

阅读原文