面试官：如果某个业务量突然提升100倍QPS你会怎么做？

原创 拾光 DotNet加油站 2025年04月09日 16:35 上海

DotNet加油站
致力于DotNet编程技术、开源项目分享，一起探索代码的无限可能。
21篇原创内容

公众号

“假设你负责的系统，某个业务线的QPS突然暴增100倍，你会怎么应对？”
——这是上周朋友去面试，被问到一道题，他答了“加机器扩容”，结果面试官眉头一皱：“如果机器不够呢？如果数据库崩了呢？”朋友当场卡壳。其实这道题就像“高压水枪”，专冲知识漏洞。

作为开发者，如果只回答“加机器”“扩容”，可能直接暴露知识盲区。**真正的答案，需要从架构设计、资源调度、容灾兜底等多个维度拆解。**

第一步：先问“为什么”，再想“怎么做”

面对突发流量，**盲目优化=挖坑埋自己**。先理清关键问题：

QPS来源是否合理？

- 是正常业务爆发（如双十一促销），还是异常流量（如恶意攻击、代码BUG）？
- 若是异常，需优先拦截（风控、限流），而非盲目扩容。

流量暴增的范围和时间？

- 是全局流量激增，还是单个接口/功能？
- 是短期高峰（如秒杀），还是长期持续？

当前系统的瓶颈在哪里？

- CPU/内存/磁盘/网络？
- 数据库？缓存？第三方服务？

第二步：分层拆解，针对性优化

快速止血：限流降级，保住核心业务

- **限流**：对非核心接口设置QPS阈值（如令牌桶算法），超限请求直接熔断。
- **降级**：关闭次要功能（如评论、推荐），确保核心链路（如支付、下单）可用。
- **预案**：提前配置好降级开关，通过配置中心实时生效。

横向扩展：无状态服务快速扩容

- **容器化+弹性伸缩**：Kubernetes自动扩缩容，应对流量波动。
- **负载均衡**：调整权重，将流量分流到压力较小的节点。
- **注意点**：确保服务无状态，避免扩容后Session丢失等问题。

缓存为王：减少穿透击穿数据库

- **本地缓存**：高频读数据（如商品信息）。
- **分布式缓存**：Redis集群抗住大部分查询请求，设置多级缓存架构。
- **缓存预热**：提前加载热点数据，避免冷启动雪崩。

数据库优化：分库分表+读写分离

- **读写分离**：主库负责写，从库集群承担读请求。
- **分库分表**：按业务拆分（用户库、订单库），或按Hash分片。
- **连接池优化**：调整最大连接数、超时时间，避免线程阻塞。

异步化：削峰填谷，解耦系统

- **消息队列**：Kafka/RocketMQ承接突发流量，后端异步消费。
- **批量处理**：合并多次请求（如库存扣减），减少数据库压力。

第三步：长期防御，构建弹性架构

全链路压测

- 定期模拟极端流量，暴露系统瓶颈（如数据库连接池耗尽、慢SQL）。
- 阿里的“全链路压测”已成为大厂标配。

监控告警体系

- 关键指标实时监控：CPU、内存、QPS、RT、错误率。
- 设置多级阈值（预警、严重、致命），通过企业微信/钉钉通知。

容灾演练

- 定期演练机房断电、网络分区、缓存崩溃等极端场景。
- 确保故障发生时，能自动切换灾备节点。

总结：高并发的本质是“分治”

应对突发流量的核心逻辑：

- **横向拆分**：用空间换时间（扩容、分库分表）。
- **纵向分层**：每层专注单一问题（缓存、异步、限流）。

🔷 **冗余设计**：假设任何环节都会挂，做好兜底方案。

如果老板要求“零预算优化”，不能加机器，你会怎么做？

欢迎评论区讨论！💡