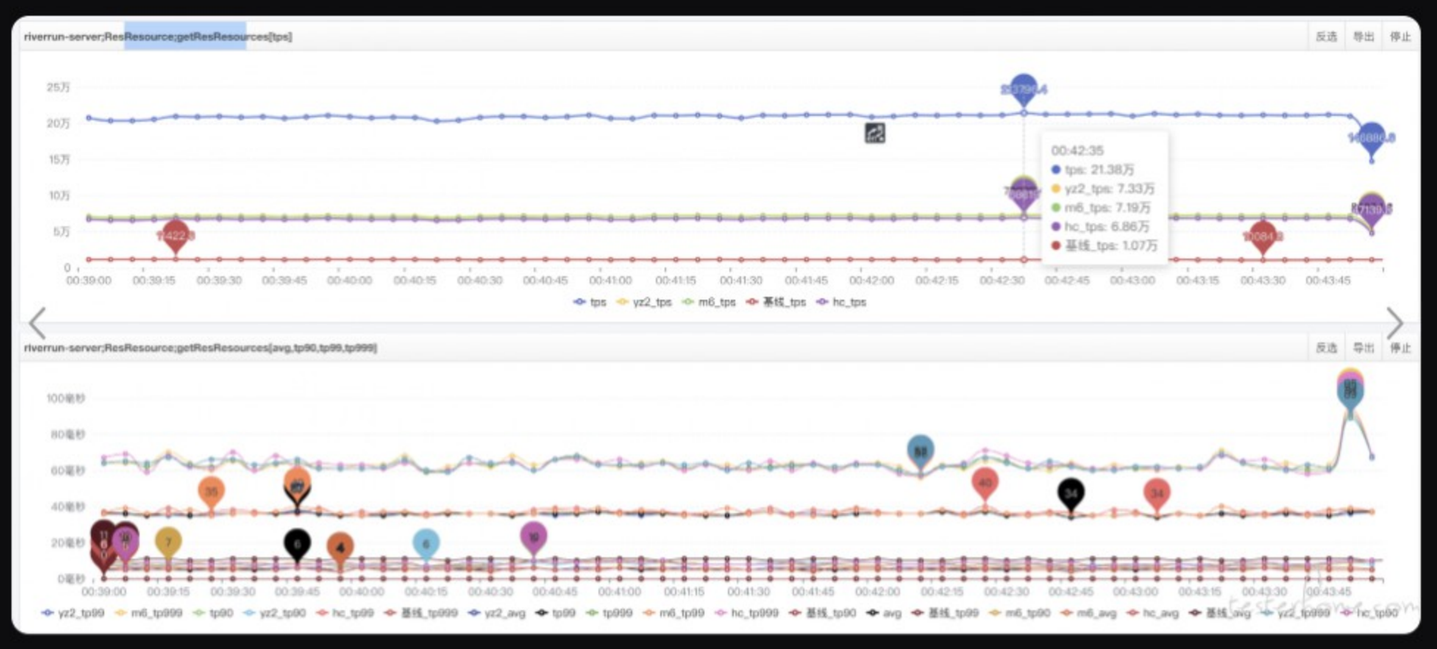
研发效能 对号入座，快看看你的应用系统用了哪些高并发技术？

京东云开发者 · 2024年04月12日 · 3317 次阅读

一 系统简介

百舸流量运营平台承接着京东金融 APP 核心资源位和京东 APP 部分重要资源位，大促单接口 QPS 达到 10w+,压测单接口到 20w+,典型的 c 端读链路高并发场景。接下来，聊聊我们的系统都有哪些应对高并发的“武功秘籍”。



二“武功秘籍”

1 缓存（redis 缓存，本地缓存）

缓存是提高系统的并发和提升系统的性能利器。redis 分布式缓存用来解决缓存容量和性能问题，本地缓存用来解决 redis 的热 key 问题和提升性能。

详情可以查看之前的文章《服务端应用多级缓存架构方案》。

2 限流

限流是保护系统的一种策略，限流是控制接受请求的速率，通过压测提前预知系统可承载的并发量，是对系统资源的前置保护，保证系统容量范围内的请求能够正常返回，超过容量的请求丢弃。

可通过 JSF 配置限流或者 sentinel 实现限流。经典算法：令牌桶，漏桶，滑动时间窗口。

3 熔断降级

熔断也是保护系统的一种手段，分布式系统中系统之间通过微服务调用，偶尔会出现依赖的某个服务不可用或者耗时骤增，导致耗尽业务线程池，从而拖垮整个服务，可通过 sentinel 配置慢调用比例或者异常比例策略，达到熔断阈值后，接下来的熔断时长内请求会自动被熔断。经过熔断时长后熔断器会进入探测恢复状态（HALF-OPEN 状态），若接下来的一个请求成功通过后，则结束熔断，否则继续熔断。 实际过程中，需要结合上下游链路，设置合理的超时时间以及兜底数据。

常见的降级类型有：日常降级；大促非核心接口降级；大促日志降级，只打印 error 级别日志。

4 异步 (CompletableFuture,MQ)

系统解耦：完成一项业务指令通常需要多个微服务协作，核心业务完成后，可通过消息的方式进行异步解耦，让其他服务订阅消息，完成各自的业务逻辑，适用于无需用户等待感知的场景。

京东云开发者
@JCloud

拥抱技术，与开发者携手创造未来！

提升性能:在 C 端用户等待的感知的场景，需要多个微服务协作，如果串行 RPC 调用，耗时是每个服务耗时之和，可通过 CompletableFuture 实现 RPC 异步调用，当使用时汇总结果，提升系统的性能。

5 池化技术

池化技术思想： 池化思想的解决的核心思想是通过预先创建数据库连接或者线程放入池中，以便在需要时可以重复使用，减少创建和销毁的开销，提高系统的性能和并发。

数据库连接池：如果是部署多台机器，注意多台机器连接数是否超过数据库最大连接数，避免出现连接不上问题。

业务线程池：自定义线程池，根据业务采用合适的拒绝策略，注意线程隔离，避免某个接口异常拖垮整个应用。

6 代码优化

减少调用链路，优化代码逻辑执行顺序，将阻断校验流程前置，优化数据结构和算法，优化查询逻辑，减少 IO 次数等。

7 JVM 调优

使用 G1 垃圾回收器，应用系统根据自己的业务情况配置 JVM 参数，常规 4C8G 通用配置可参考：

–Xms4096m –Xmx4096m –XX:MaxMetaspaceSize=256m –XX:MetaspaceSize=256m –XX:+UseG1GC –XX:MaxGCPauseMillis=80
核心参数：–Xms 初始堆大小，–Xmx 最大堆大小，MaxMetaspaceSize 最大元空间大小，MetaspaceSize 表示 Metaspace 首次使用不足时触发 Full GC（全面垃圾回收）的阈值，垃圾回收机制使用 G1 回收器，MaxGCPauseMillis 在 jvm 垃圾回收过程中允许停顿的最大毫秒时间。

8 分治思想，横向扩展，

应用服务应该设计为无状态的，可通过增加应用实例数量来应对突发流量，将流量分到每台机器上，同样可以将应用进行按照业务拆分，单独部署，提高系统并发。

合并批量请求，将多次调用改为一次批量调用，减少网络开销。

9 预热

通过定时任务或者初始化脚本提前将数据加载到内存，提高系统的性能，常见的有缓存数据预热，ES 数据预热等。

针对应用升级或者重启抖动，可以通过 JSF 预热的方式，应用重启后，在预热时间内，流量逐渐增加的方式，减少抖动。

JSF 预热可参考文章《后端服务之应用预热》

10 数据异构

业务数据通常存储在支持事务的关系型数据库中，当在面对复杂查询场景时捉襟见肘，可将数据通过 binlog 异构到 ES 中，ES 支持复杂场景的查询并且有较高的性能，轻松突破数据库单表数据量大及多表关联查询瓶颈。

数据异构可参考文章《记一次生产慢 sql 索引优化及思考》中的目录五：长期优化方案。

11 分库分表，数据库优化

分库和分表各抗什么？

分表：当一个表中的数据量过大时，会导致查询速度变慢、插入和更新操作效率下降等问题。通过分表，每个小表的数据量就相对较小，性能问题得以缓解。

分库：当一个数据库实例无法承受大量数据的存储和并发时，可通过分库来分散系统压力。

通常情况下，分库和分表是结合使用的。

数据库优化中常见的是 sql 优化，是否命中索引，提高服务器硬件配置。

三 总结

以上为百舸系统处理高并发问题的一些策略，高并发架构是演进而来，避免过度设计，没有一个技术能解决所有的问题，抓住关键矛盾，使用前一定要做好调研和评估，还有哪些？欢迎补充。

3 个赞

暂无回复。

需要 登录 后方可回复, 如果你还没有账号请点击这里 注册 。