# 一、会话跟踪技术

主要分为以下几种

1.Cookie技术

Cookie技术是一种在客户端保持会话跟踪的解决方案，会话数据保存在客户端浏览器。

Cookie在用户第一次访问服务器时，由服务器通过响应头的方式发送给客户端浏览器；当用户再次向服务器发送请求时会附带上这些文本信息。

2.Session技术

Session是指使用HttpSession对象实现会话跟踪的技术，是一种在服务器端保持会话跟踪的解决方案。

3.URL重写技术

URL重写即使浏览器不支持 cookie 或在用户禁用 cookie 的情况下，这种方案也能够工作。

4.隐藏表单域技术

HTML 表单中可以含有如下的条目：

这个条目的意思是：在提交表单时，要将指定的名称和值自动包括在 GET 或 POST 数据中。这个隐藏域可以用来存储有关会话的信息，

但它的主要缺点是：仅当每个页面都是由表单提交而动态生成时，才能使用这种方法。

# 二、JVM生产环境具体优化？

1.JDK1.8+优先使用G1收集器，可以摆脱各种烦恼。

java -jar -XX：+UseG1GC -Xms2G -Xms2G -Xss256k

-XX：MaxGCPauseMillis=300

-Xloggc：/logs/gc.log -XX：+PrintGCTimeStamps -XX：PrintGCDetails test.jar

I、-Xms与-Xmx设置相同，减少动态内存调整所带来的内存交换

评估-Xmx方法：第一次起始设计大一点，跟踪监控日志，调整为堆峰值*2~3即可

II、最多300ms的STW的时间，在200~500区间即可，增大时可减少GC次数，提高吞吐

III、-Xss设置为128k/256k。JDK1.5以后虚拟机栈默认每个线程分配1M空间，有些多余，若处理线程过多，内存压力很大。

同时业务不涉及复杂运算，128K够用。涉及复杂运算256K也够用。所以超过256K就要考虑优化，不建议超过256k。

IV、G1一般不设置新生代的大小-Xmn，因为G1新生代是动态调整的。

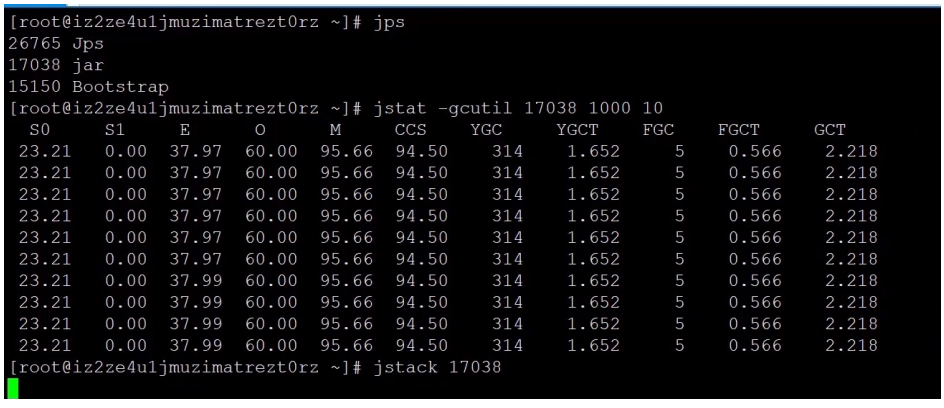
# 三、线上OOM具体排查？

一、

1.jps

2.jstat -gcutil pid 1000 10

三、jmap自dump



3.使用jstack太难看懂

二、可以安装使用arthas

4.直接通过java -jar命令启动arthas，在向导中选择你想监控的进程号。

5.dashboard命令可以通过仪表盘的方式动态的在线展示jvm的运行情况。

6.如果想看详细信息，例如当前内存中哪个对象没有被正常释放，它占用空间是多少，堆栈信息等

可以使用与jmap相似的一个命令heapdump去把堆中所有信息dump到某个文件中

heapdump /tmp/dump.hprof

7.用xftp把dump文件下载到本地的某个目录，通过visualVM的离线分析功能去对dump文件可视化分析

# 四、如何全表扫描一个大表？

全表扫描在数据库上优化的余地很小，完全是看你系统的IO的能力。

1使用多台机器并行扫描一张表

但是这样需要对整个表加锁，并不适合在并发量很大的场景使用

2.表分区

通过一致性hash进行分区，然后通过多线程进行扫描。

3.钞能力，高端存储。

# 五、业务层如何实现高可用？

1.扩展

扩展是最常见的提升系统可靠性的方法，系统的扩展可以避免单点故障。

1. 垂直扩展：是在同一逻辑单元里添加资源从而满足系统处理能力上升的需求。比如，当机器内存不够时，我们可以帮机器增加内存，或者数据存不下时，我们为机器挂载新的磁盘。

2. 水平扩展：通过增加一个或多个逻辑单元，并使得它们像整体一样的工作。

在实际应用中，水平扩展最常见：

通常我们在部署应用服务器的时候，都会部署多台，然后使用 nginx 来做负载均衡，nginx 使用心跳机制来检测服务器的正常与否，无响应的服务就从集群中剔除。

这样的集群中每台服务器的角色是相同的，同时提供一样的服务。

2.限流

一个系统的处理能力是有上限的，当服务请求量超过处理能力，通常会引起排队，造成响应时间迅速提升。

如果对服务占用的资源量没有约束，还可能因为系统资源占用过多而宕机。

常见的限流算法有：漏桶、令牌桶、滑动窗口计数。

1漏桶算法可以使用 Redis 队列来实现，生产者发送消息前先检查队列长度是否超过阈值，超过阈值则丢弃消息，否则发送消息到 Redis 队列中；

消费者以固定速率从 Redis 队列中取消息。Redis 队列在这里起到了一个缓冲池的作用，起到削峰填谷、流量整形的作用。

2Guava 中的限流工具 RateLimiter，其原理就是令牌桶算法

3.降级

通过开关控制，降级部分非主流程的业务功能，减轻系统依赖和性能损耗，从而提升集群的整体吞吐率。

降级的典型应用是：电商活动期间关闭非核心服务，保证核心买买买业务的正常运行。

降级和熔断可用hytrix。

4.熔断

在分布式系统中，如果调用的远程服务或者资源由于某种原因无法使用时，没有这种过载保护，就会导致请求阻塞在服务器上等待从而耗尽服务器资源。

很多时候刚开始可能只是系统出现了局部的、小规模的故障，然而由于种种原因，故障影响的范围越来越大，最终导致了全局性的后果。即避免级联故障。

熔断器的基本原理，包含三个状态：

1服务正常运行时的 Closed 状态，当服务调用失败量或失败率达到阈值时，熔断器进入 Open 状态

2在 Open 状态，服务调用不会真正去请求外部资源，会快速失败。

3当进入 Open 状态一段时间后，进入 Half-Open状态，需要去尝试调用几次服务，检查故障的服务是否恢复。如果成功则熔断器关闭，如果失败，则再次进入 Open 状态。

5.灰度发布