# Jvm

当出现了内存溢出如何定位排查

Jmap –dump命令生成堆内存快照 jmap-heap命令显示java堆详细信息。如使用哪种回收器，参数配置。分代状况等

Jhat 工具分析。或者使用ibm heapanalyzer 或eclipse memomry analyzer分析

Jvm调优

# Tomcat以及服务器

Tomcat如何调优

<https://blog.csdn.net/qq_28109171/article/details/84256783>

并发优化

Connector 优化

Connector是连接器，负责接收客户的请求，以及向客户端回送响应的消息。所以 Connector的优化是重要部分。默认情况下 Tomcat只支持200线程访问，超过这个数量的连接将被等待甚至超时放弃，所以我们需要提高这方面的处理能力。

参数说明

maxThreads 客户请求最大线程数

minSpareThreads Tomcat初始化时创建的 socket 线程数

maxSpareThreads Tomcat连接器的最大空闲 socket 线程数

enableLookups 若设为true, 则支持域名解析，可把 ip 地址解析为主机名

redirectPort 在需要基于安全通道的场合，把客户请求转发到基于SSL 的 redirectPort 端口

acceptAccount 监听端口队列最大数，满了之后客户请求会被拒绝（不能小于maxSpareThreads ）

connectionTimeout 连接超时

minProcessors 服务器创建时的最小处理线程数

maxProcessors 服务器同时最大处理线程数

URIEncoding URL统一编码

开启线程池

配置

参数说明

name：线程池名称，用于 Connector中指定。

namePrefix：所创建的每个线程的名称前缀，一个单独的线程名称为 namePrefix+threadNumber。

maxThreads：池中最大线程数。

minSpareThreads：活跃线程数，也就是核心池线程数，这些线程不会被销毁，会一直存在。

maxIdleTime：线程空闲时间，超过该时间后，空闲线程会被销毁，默认值为6000（1分钟），单位毫秒。

maxQueueSize：在被执行前最大线程排队数目，默认为Int的最大值，也就是广义的无限。除非特殊情况，这个值不需要更改，否则会有请求不会被处理的情况发生。

prestartminSpareThreads：启动线程池时是否启动 minSpareThreads部分线程。默认值为false，即不启动。

threadPriority：线程池中线程优先级，默认值为5，值从1到10。

className：线程池实现类，未指定情况下，默认实现类为org.apache.catalina.core.StandardThreadExecutor。如果想使用自定义线程池首先需要实现 org.apache.catalina.Executor接口。

tomcat各参数的意义

-server：启用 JDK的 server 版本；

-Xms：Java虚拟机初始化时堆的最小内存，一般与 Xmx配置为相同值，这样的好处是GC不必再为扩展内存空间而消耗性能；

-Xmx：Java虚拟机可使用堆的最大内存；

-XX:PermSize：Java虚拟机永久代大小；

-XX:MaxPermSize：Java虚拟机永久代大小最大值；

# 分布式

一致性hash

hash环

单调性(Monotonicity)，单调性是指如果已经有一些请求通过哈希分派到了相应的服务器进行处理，又有新的服务器加入到系统中时候，应保证原有的请求可以被映射到原有的或者新的服务器中去，而不会被映射到原来的其它服务器上去。 这个通过上面新增服务器ip5可以证明，新增ip5后，原来被ip1处理的user6现在还是被ip1处理，原来被ip1处理的user5现在被新增的ip5处理。

分散性(Spread)：分布式环境中，客户端请求时候可能不知道所有服务器的存在，可能只知道其中一部分服务器，在客户端看来他看到的部分服务器会形成一个完整的hash环。如果多个客户端都把部分服务器作为一个完整hash环，那么可能会导致，同一个用户的请求被路由到不同的服务器进行处理。这种情况显然是应该避免的，因为它不能保证同一个用户的请求落到同一个服务器。所谓分散性是指上述情况发生的严重程度。好的哈希算法应尽量避免尽量降低分散性。 一致性hash具有很低的分散性

平衡性(Balance)：平衡性也就是说负载均衡，是指客户端hash后的请求应该能够分散到不同的服务器上去。一致性hash可以做到每个服务器都进行处理请求，但是不能保证每个服务器处理的请求的数量大致相同，如下图

服务器ip1,ip2,ip3经过hash后落到了一致性hash环上，从图中hash值分布可知ip1会负责处理大概80%的请求，而ip2和ip3则只会负责处理大概20%的请求，虽然三个机器都在处理请求，但是明显每个机器的负载不均衡，这样称为一致性hash的倾斜，虚拟节点的出现就是为了解决这个问题。

当服务器节点比较少的时候会出现上节所说的一致性hash倾斜的问题，一个解决方法是多加机器，但是加机器是有成本的，那么就加虚拟节点，比如上面三个机器，每个机器引入1

常见的hash算法

Paxos算法

分布式一致性

Paxos将系统中的角色分为提议者 (Proposer)，决策者 (Acceptor)，和最终决策学习者 (Learner):

Proposer: 提出提案 (Proposal)。Proposal信息包括提案编号 (Proposal ID) 和提议的值 (Value)。

Acceptor：参与决策，回应Proposers的提案。收到Proposal后可以接受提案，若Proposal获得多数Acceptors的接受，则称该Proposal被批准。

Learner：不参与决策，从Proposers/Acceptors学习最新达成一致的提案（Value）。

<https://zhuanlan.zhihu.com/p/31780743>

https://zhuanlan.zhihu.com/paxos

# 消息队列

MQ如何保证消息不丢失

1. 突然宕机

通过消息持久化

1. ack确认机制

消费者消费过后要回复一个确认(超时没有响应别的消费者就回消费)。但是会可能产生重复消费

集群

一台宕机了换从机

# http相关

301 302区别

301永久重定向 302临时重定向