# 性能测试基础指标

## 1.1、并发用户数在测试中的意义

Jmeter在运行过程中使用java线程模拟用户的行为，每一个线程都对应一个真正的用户，那么，在同一时刻正在执行操作的用户数量就是并发用户数。这个数值可以通过C=nl/T这个公式算出来，C就是并发用户数，n是在T时间段内进行操作的用户的个数，l是每个用户操作的时长。通过上面的公式可以看出，并发用户数这是一个统计值，如果需要准确的估算出这个值需要准确的统计n和l。

对于简单的负载测试场景，在这个场景中，并发用户数一经设置就不再改变，那么确定了并发用户数就等于确定了测试场景。

## 1.2、对cpu并发的理解

简单说smp系统支持的并发个数就等于top中看到的cpu核心数，但是在操作系统的支持下，通过合适的算法我们可以让机器能支持多于核心数的并发任务量。实现这种功能的核心就是对空闲时间的再利用。但是要注意，当cpu的时间没有空余的时候，cpu的处理能里就不能再提高了，这也就是我们常说的到瓶颈了。

## 1.3、TPS和平均返回时间

通过经典的tps公式“TPS=并发用户数/平均返回时间”可以看出，tps和平均返回时间是一体两面的关系，在并发数一定的时候他们都反应系统处理任务的能力。这两个指标本身非常好理解，不过在使用与计算方法上却很值得细说。

在分析问题的时候经常使用瞬时的tps和返回时间，但是出报告的时候tps和返回时间却表示了测试过程中系统的整体状态，那么这两个阶段的指标是不同的么？其实是不是的，tps的计算公式可以写成这样：tps=完成任务数/T，平均返回时间的计算公式是这样的：平均返回时间=(任务1的返回时间+任务2的返回时间+...+任务n的返回时间)/n。通过公式就可以看到，在分析问题的时候使用的tps和返回时间的采样间隔很短，也就是T的时间段很短，相应的n这个完成任务的数量就会少，那么根据数学上极限的思想我们就可以把这个短时间的平均值看作瞬时值来分析问题，但是报告里要体现系统在测试过程中的总体情况，所以T就会等于整个测试场景运行的时间。所以分析问题和写报告时用的tps和返回时间是一个概念。

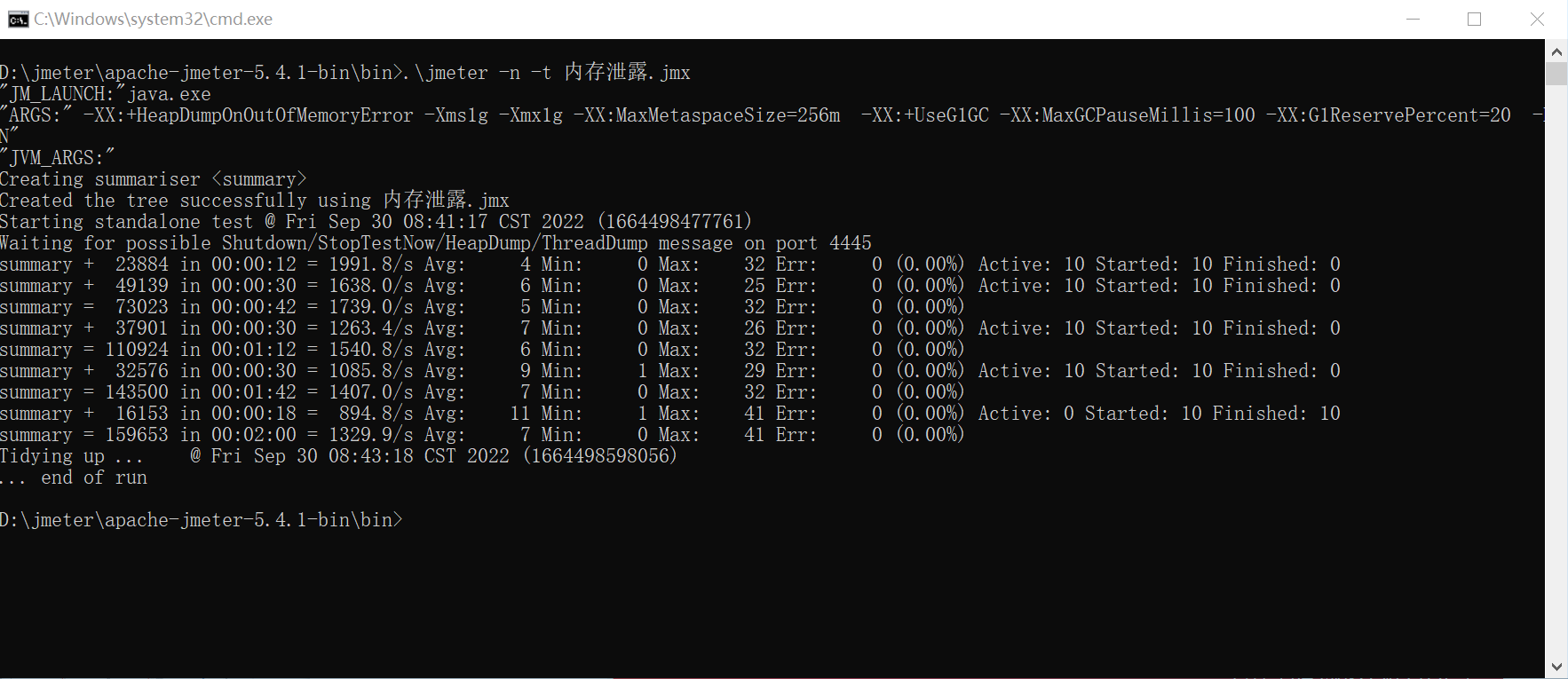
## 1.4、性能测试指标

现行性能测试重要的指标基本有三个，分别是tps、平均返回时间和95百分位相应时间。

这里解释一下“95百分位相应时间”这个指标，这个指标是常用来判断系统响应时间稳定性的指标。假设我们在一段时间内执行了100个任务，按返回时间的长短由低到高排序，那么从第一个返回时间最短的任务开始数，数到第95个任务，这个任务的返回时间就是所谓的“95百分位相应时间”。与其相似的还有“90百分位相应时间”和“99百分位相应时间”，他们的含基本相同。

这里再解释下为什么有平均值了还要加一个95百分位的指标。我们在计算方差和标准差的时候都会把单项的值减去平均值，这个操作的含义就是通过单项值和平均值的差值来反应统计指标的离散程度。但是在性能测试的场景里，小于平均值的返回时间是优秀的返回时间，他们不是分析问题的点，所以我们在分析问题的时候基本不会关注这些点，所以我们只关注平均值以上的离散情况，所以就会有“95百分位相应时间”这个指标了，这个指标里平均值越近，说明系统稳定性越好。

# 可疑测试结果举例



上图的测试结果是最可疑的一类测试结果，在测试的过程中，tps缓慢下降，但是总体测试结果是通过的，一般导致这种情况的根本原因是后台的业务数据随时间积压，比如先增加后查询的测试场景。并且在很多时候这种测试结果会是非常严重的性能问题的前兆，如内存泄漏。所以在看到这种场景以后需要加压复测并观察jvm内存使用情况来确定问题。

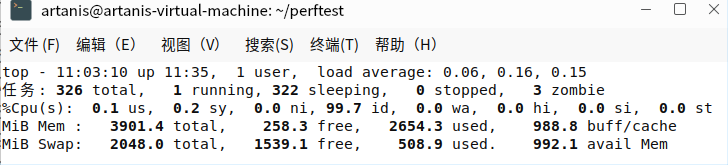
在对jemter的nogui模式的测试结果做一下介绍，除去一眼就能看出作用的几列，这里着重说明表示请求返回时间离散程度的avg，max和min三列。首先，max和min表示了整个测试样本的波动范围，而avg列表示了平均值轴线在哪，大部分时候avg列的值都是偏向min列的，随着并发数的增长，排队情况开始增加，avg列的值就会离min列越来越远，并且max列的值也会增大，在后台观测的时候也会看到除了cpu利用率增长了以外，loadaverage也会增加。

上面介绍了jmeter的nogui模式下的测试结果，但是jmeter并不止这一个测试结果，通过jtl文件生成的html报告才是分析问题的最终依据，通过折线图展示测试结果也更直观。但是分析折线图的基础还是对这些基础指标的理解。

# top监控工具介绍

top工具是最常用的监控工具，所以最后还是来介绍一下top这个工具。top工具是procps项目下的一个子工具，从这个项目名就可以看出，top工具的数据来源是proc文件系统，这里面的数据都是Linux内核负责收集的，top只是把他们组织起来而已。

top的展示的部分有两部分，一部分是头，这部分展示的都是系统的综合信息，这部分信息来源是”/proc/”，这下面显示的都是操作系统的整体信息，头部分如下所示：



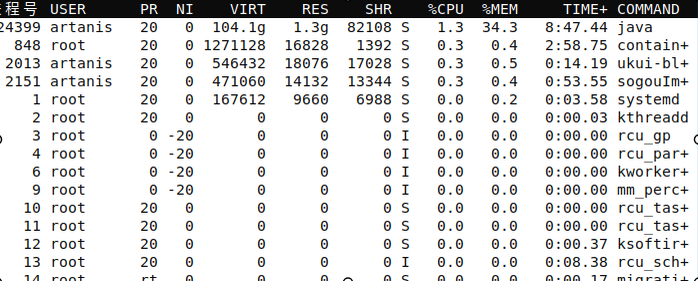
图T-1

图T-1显示了完整的top头部分，这部分的数据主要来自/proc/stat和/proc/status两个问文件，而load avgrage的信息来自于/proc/loadavg文件。我们从上到下介绍一下重要的指标。

load average：这个指标是非常重要的一个指标，它是操作系统根据当前系统的内TASK\_RUNNING状态进程的数量算出来的，它表面的意思也就时反应当前系统正在处理的任务数，没别的意思。与它相对的还有一个概念叫cpu负载，这个cpu负载是内核给smp系统做cpu负载均衡用的，这两个概念千万不能混淆。

buff/cache：这个显示系统的io缓存。我们在调用wirte函数写文件的时候，在一般情况下，写磁盘的操作并不会发生在write函数的上下文中，write函数只修改内存里映射了文件内容的内存块的内容然后就返回了，而修改写入磁盘的操作由操作系统在合适的时候写入磁盘。读缓存更简单，就是只有在第一次读的时候进行磁盘读，后续如果再读到这块内容就直接返回这个块里的数据。

swap：交换分区并不是内存的扩展，操作系统在内存紧张的时候会将合适的页面的内容写到交换分区的页面里，这样就能把腾出的空间用于其他工作。但是被交换出去的页面是被写到了磁盘上，这就导致了被交换出去的页面是不能直接被cpu寻址的，所以cpu如果寻址的是swap分区里的块也一样会触发缺页中断，只不过处理方式和传统的缺页中断不一样，传统的匿名页和文件映射页面的缺页中断会分配新的页面和从文件中读取内容到页面，swap的缺页中断会把页面从交换分区里换回内存里。由于swap块在换出的过程中也是刷新了页表的，所以换入操作和传统的缺页异常的资源消耗是差不多的。



图T-2

这里是从进程的角度来监控资源的消耗情况，这些数值的来源是/proc/<pid>/stat、/proc/<pid>/status、/proc/<pid>/statm等进程文件。

PR和NI：这两列都表示进程的优先级，所以放在一起说，PR指的是实时进程的优先级，这个优先级是给实时进程用的，比如终端进程等交互进程就是实时进程。我们一般创建的进程都是普通进程，应用的优先级是NI，我们熟悉的CFS调度算法里的虚拟时间片就是根据这玩意算出来的。

VIRT、RES和SHR：它们都是内存指标，所以放一起说。VIRT显示的是进程申请的内存。C语言里申请和初始化是两个操作，在我们malloc内存的时候操作系统实际上并没有给申请者任何内存，如果我们不使用memset等函数初始化就直接使用malloc返回的指针，一样会报段错误的。所以，malloc了的内存是虚假的内存，用VIRT表示，memset初始化过的才是真正分配了内存的，用RES表示。最后SHR是进程使用的共享内存，这个基本就是进程动态连接占用的内存。