

# JMeter性能测试

## 1 JMeter常见插件

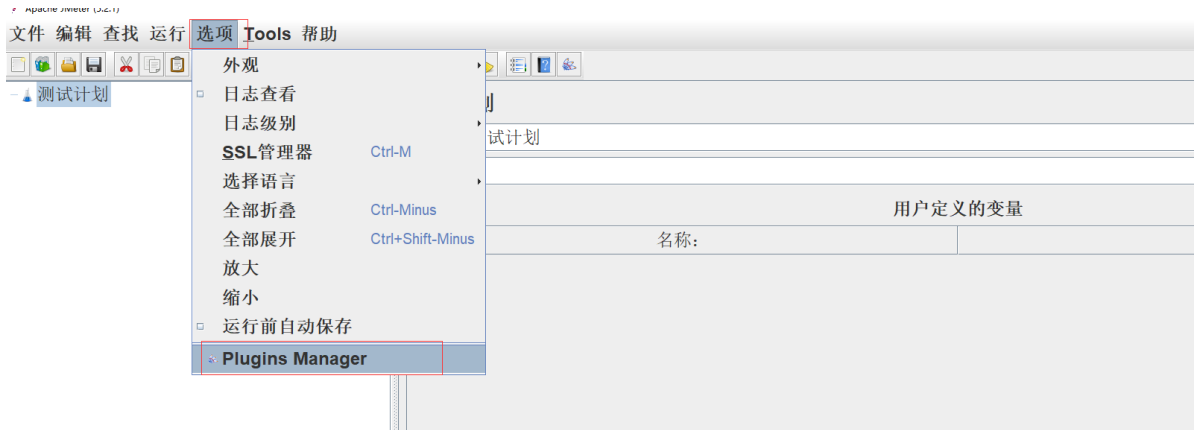
Jmeter安装插件的介绍网址：<https://jmeter-plugins.org/install/Install/>

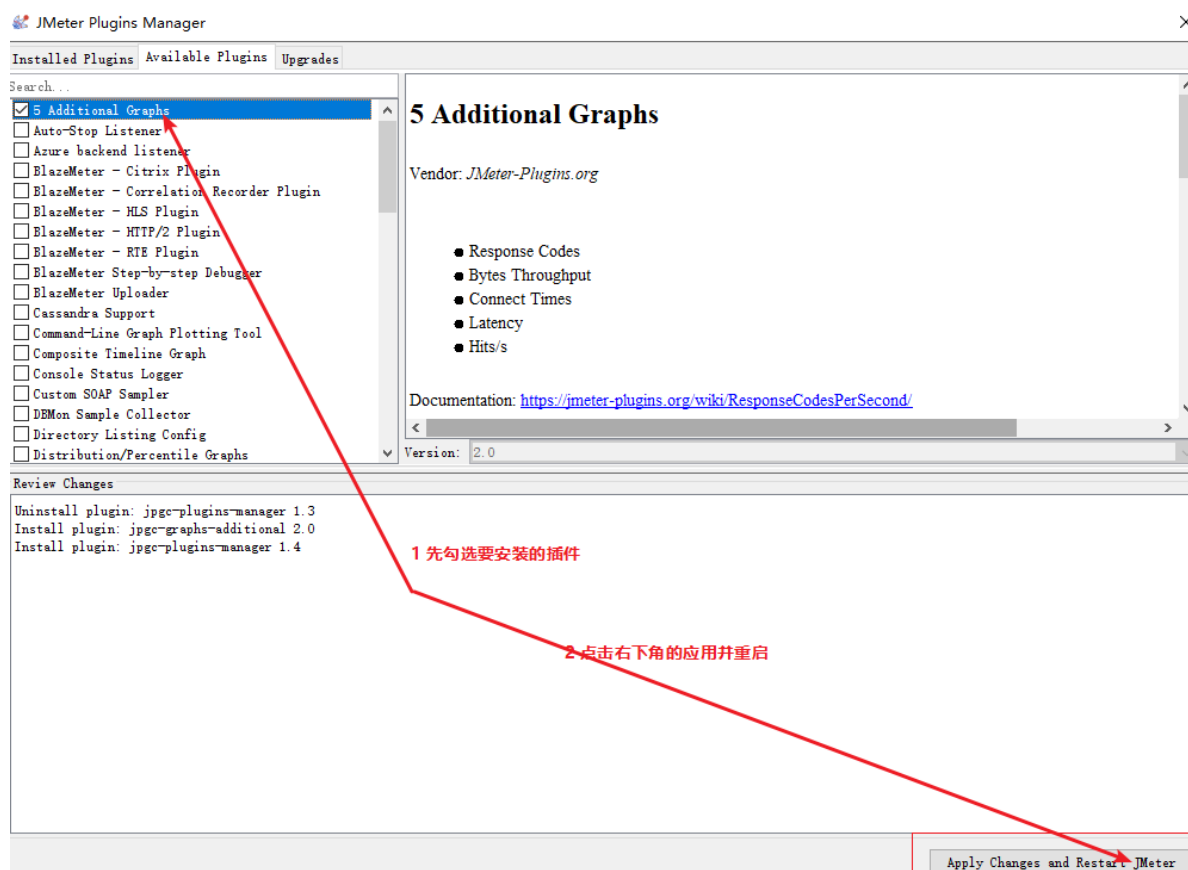
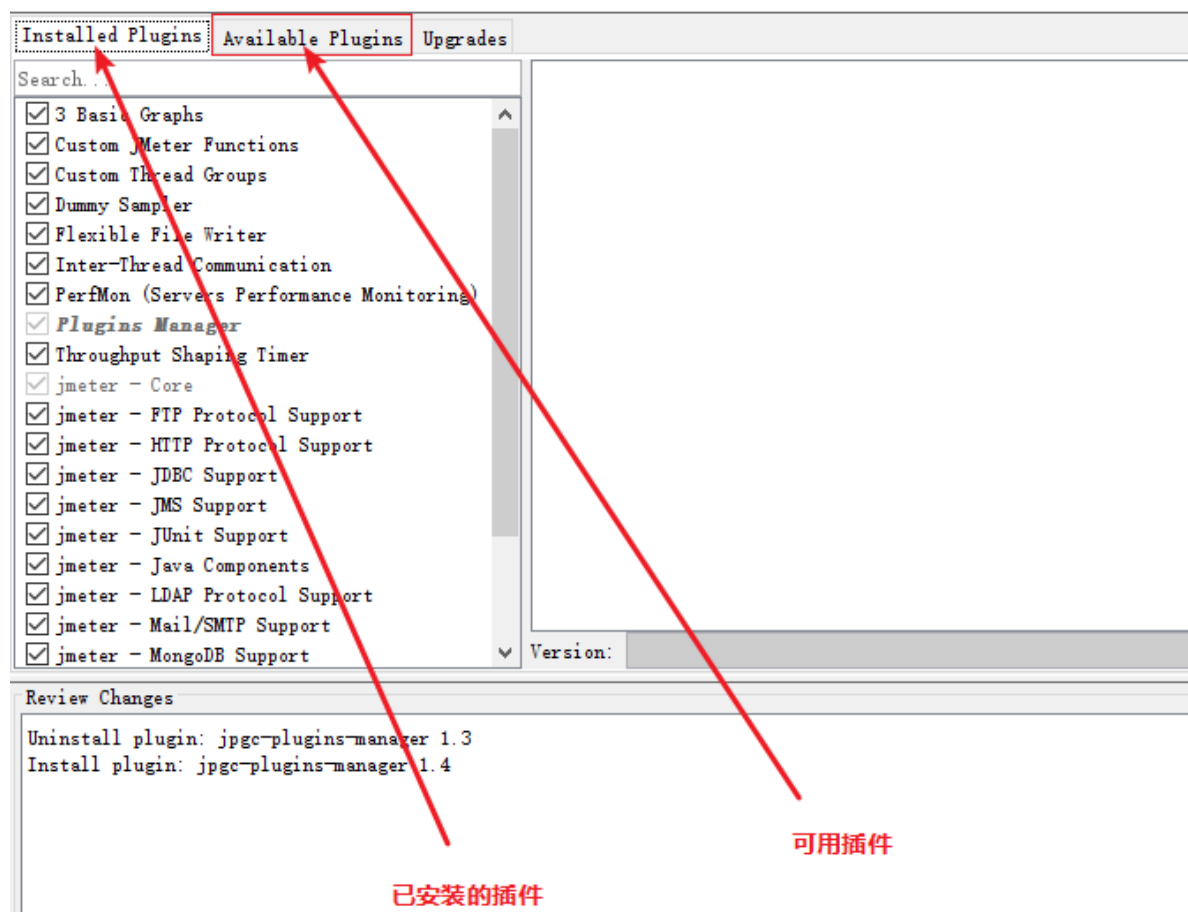
在网站中，按照提示下载jmeter插件jar包

把jmeter-plugins-manager-1.6.jar放在jmeter的lib/ext目录当中

本地磁盘 (D:) > developer tools > apache-jmeter-5.2.1 > lib > ext			
名称	类型	大小	
ApacheJMeter_bolt.jar	Executable Jar File	17 KB	
ApacheJMeter_components.jar	Executable Jar File	708 KB	
ApacheJMeter_core.jar	Executable Jar File	1,664 KB	
ApacheJMeter_ftp.jar	Executable Jar File	15 KB	
ApacheJMeter_functions.jar	Executable Jar File	116 KB	
ApacheJMeter_http.jar	Executable Jar File	475 KB	
ApacheJMeter_java.jar	Executable Jar File	45 KB	
ApacheJMeter_jdbc.jar	Executable Jar File	57 KB	
ApacheJMeter_jms.jar	Executable Jar File	95 KB	
ApacheJMeter_junit.jar	Executable Jar File	21 KB	
ApacheJMeter_ldap.jar	Executable Jar File	48 KB	
ApacheJMeter_mail.jar	Executable Jar File	59 KB	
ApacheJMeter_mongodb.jar	Executable Jar File	26 KB	
ApacheJMeter_native.jar	Executable Jar File	14 KB	
ApacheJMeter_tcp.jar	Executable Jar File	30 KB	
EventSupport.jar	Executable Jar File	4 KB	
jmeter-plugins-graphs-additional-2.0.jar	Executable Jar File	11 KB	
jmeter-plugins-manager-1.6.jar	Executable Jar File	885 KB	
readme.txt	文本文档	1 KB	

重启Jmeter，让jar重新加载生效





## 2 JMeter性能测试报告

聚合报告

测试计划->右键->监听器->聚合报告



1. Label: 每个请求的名称 (勾选: 在标签中包含组名称, 显示线程组名-取样器名)
2. 样本: 各请求发出的数量
3. 平均值: 平均响应时间 (单位: 毫秒)。默认是单个Request的平均响应时间
4. 中位数: 中位数, 50% <= 时间
5. 90%百分比: 90% <= 时间
6. 95%百分比: 95% <= 时间
7. 99%百分比: 99% <= 时间
8. 最小值: 最小响应时间
9. 最大值: 最大响应时间
10. 异常%: 请求的错误率 = 错误请求的数量/请求的总数
11. 吞吐量: 吞吐量。默认情况下表示每秒完成的请求数, 一般认为它为TPS。
12. 接收 KB/sec: 每秒从服务器端接收到的千字节数
13. 发送 KB/sec: 每秒向服务器发送的千字节数

案例场景:

循环次数: 永远

持续时间: 60s

访问百度, 汇总聚合报告数据



线程组

名称: 线程组

注释:

在取样器错误后要执行的动作

☒ 继续 ☐ 启动下一进程循环 ☐ 停止

线程属性

线程数: 1

Ramp-Up时间(秒): 1

循环次数 ☒ 永远 ☐

☐ 延迟创建线程直到需要

☒ 调度器

调度器配置

⚠ If Loop Count is not -1 or Forever, duration will be min(Duration, Loop Count \*  
持续时间(秒) 60

启动延迟(秒)



HTTP请求

名称: HTTP请求

注释:

基本 高级

Web服务器

协议: 服务器名称或IP: www.baidu.com

HTTP请求

方法: GET 路径:

☐ 自动重定向 ☒ 跟随重定向 ☒ 使用 KeepAlive ☐ 对POST使用multipart / form-data ☐ 与浏览器兼容的头

参数 消息体数据 文件上传

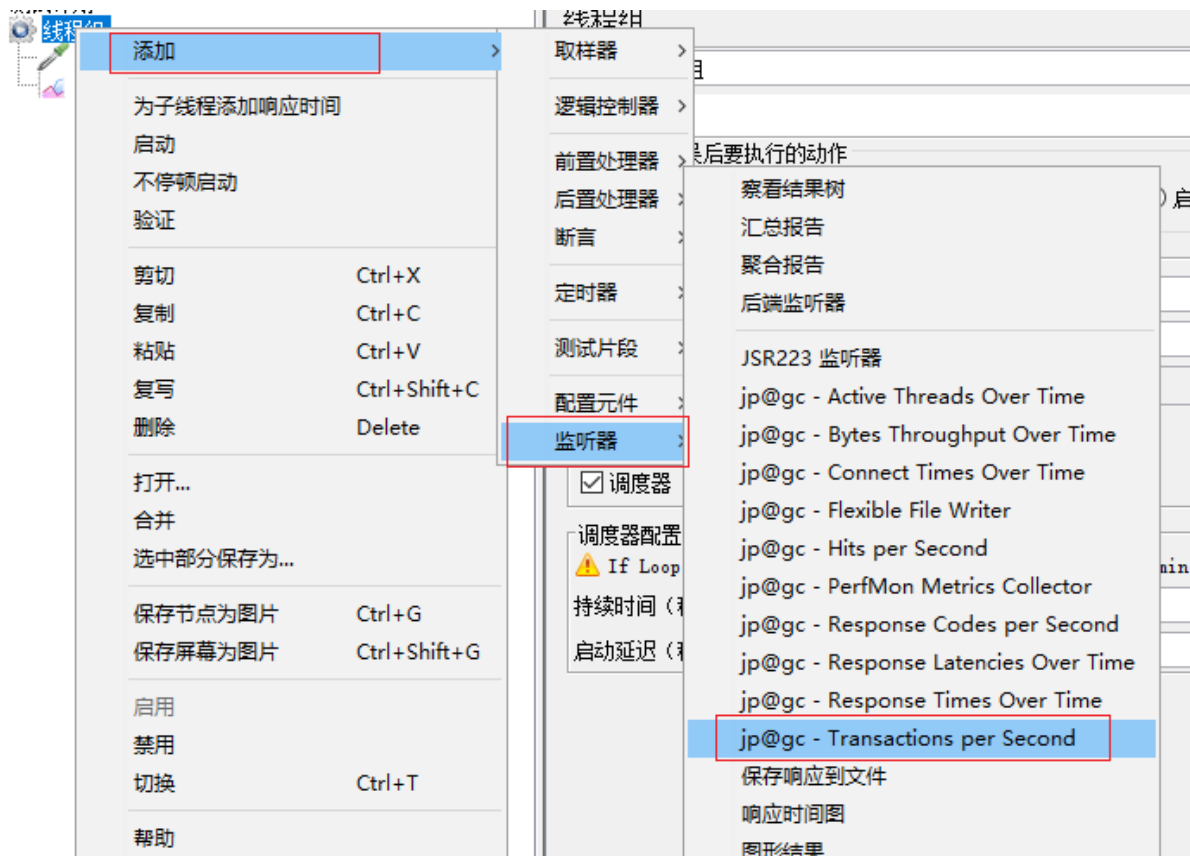
同请求一起发送参数:

名称:	值	编码?
-----	---	-----

## 3 JMeter常见图表

### 3.1 查看TPS图表

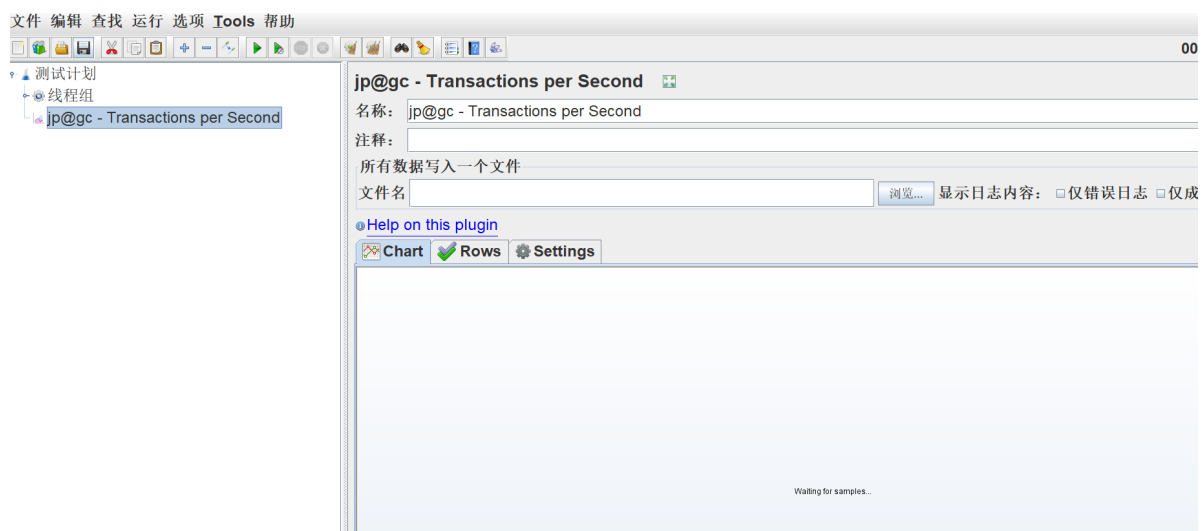
添加 -- 》监听器 -- 》 Transactions per Second



## 语法

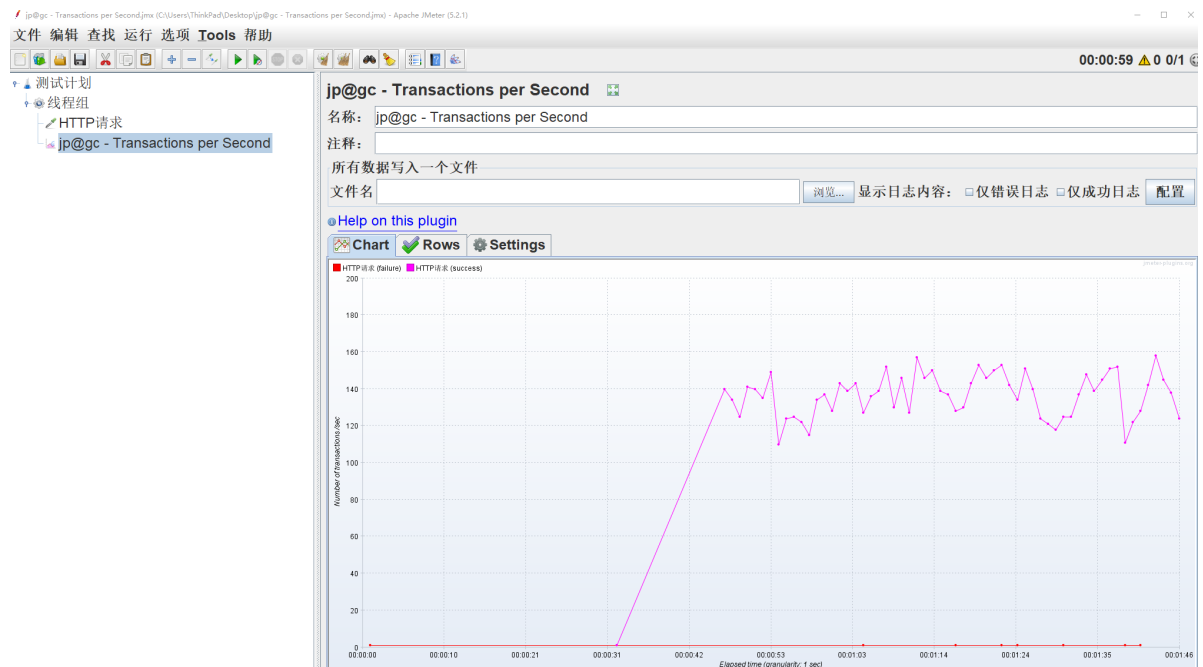
工作原理：运行后，可以每隔一段时间，采集一组数据，把这些数据用直线连接起来，从而形式一个曲线图，就是我们看到的TPS变化的曲线图。

必须运行性能测试脚本才能采集



## 运行

运行单用户连续访问百度首页的脚本，运行60秒，可以得出下图。



## 注意实现

TPS的曲线图精度最小是1，如果TPS低于1请求数每秒，那么显示还是1。

Jmeter聚合报告显示TPS是30请求数每分钟，结果TPS显示为1。

为什么TPS是30请求数每分钟时，在TPS图表中显示是1？

答案是因为：TPS的图表单位是秒，然后聚合报告显示的是30请求数每分钟时需要换算成秒才行。换算成秒时，TPS是0.5/s低于1，所以显示1

## 3.2 响应时间

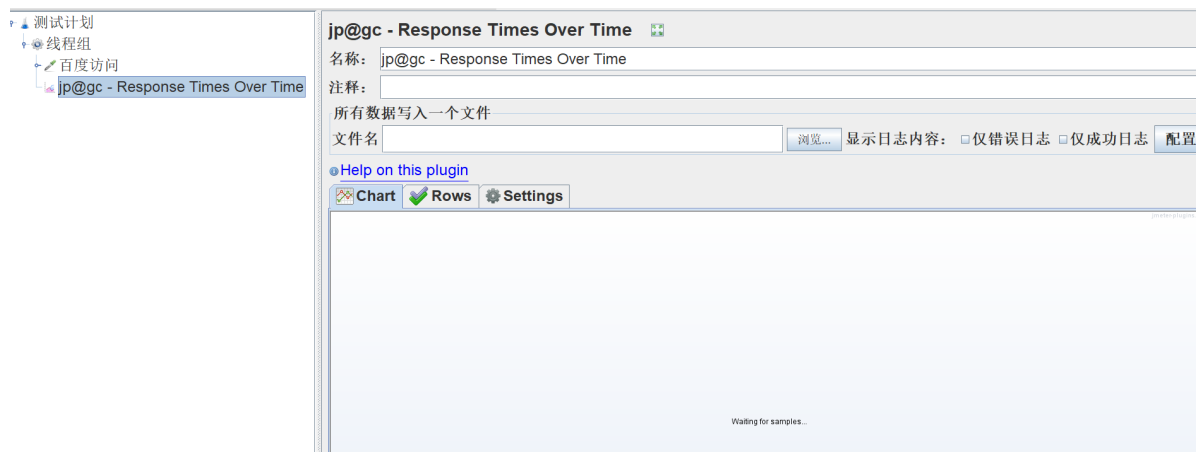
Jmeter的聚合报告已经能够显示响应时间，但是不会显示每秒的时间

查看响应时间的图表，可以显示进行性能测试时，每秒的响应时间，从而得到响应时间的变化曲线图

### 语法

可以不用修改任何配置，直接使用默认配置即可统计响应时间

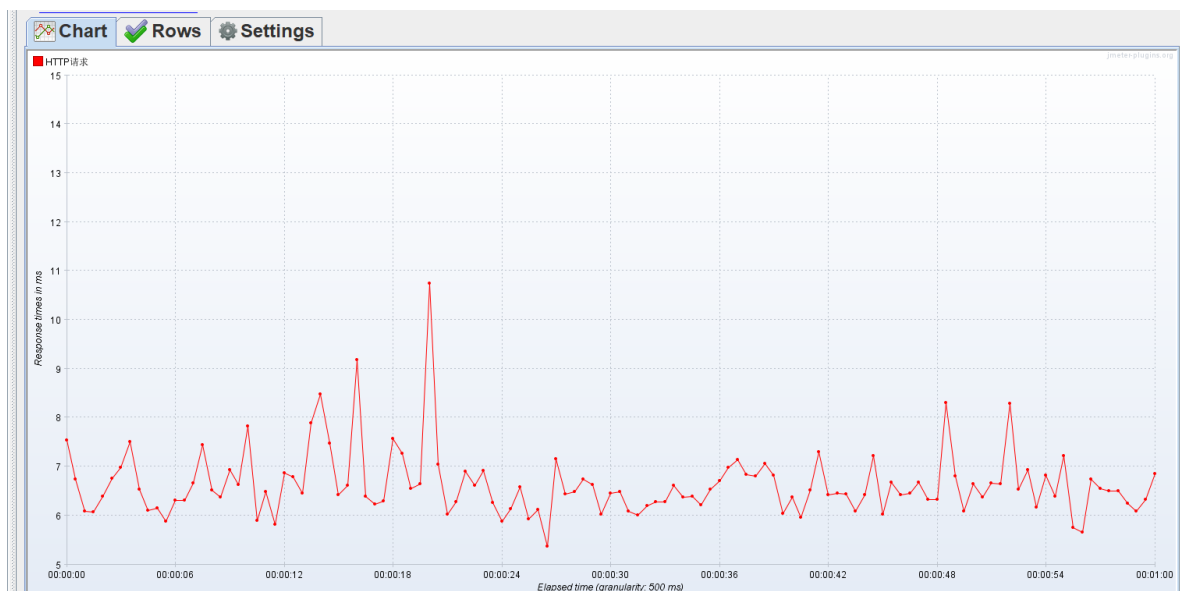
也是需要经过运行才能收集到数据，查看到曲线图



## 运行

运行后，可以看到下图中单用户连续请求的响应时间

理论上：单用户连续并发的响应时间应该很平稳，如果有波动，就证明服务器在架构设计、网络等环节可能存在问题。



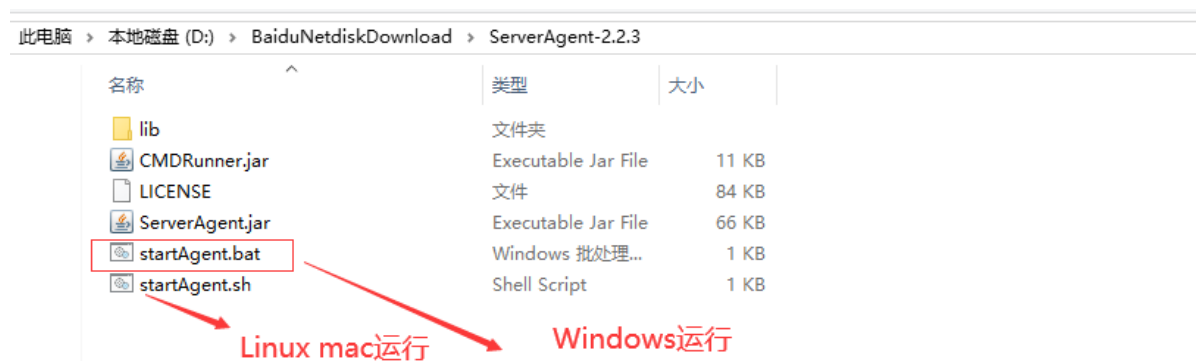
## 3.3 服务器资源使用率

通过第三方插件来监控服务器的资源使用率（CPU、内存、网络、硬盘）

借助Jmeter插件ServerAgent来监控服务器CPU等资源使用率

### serverAgent使用

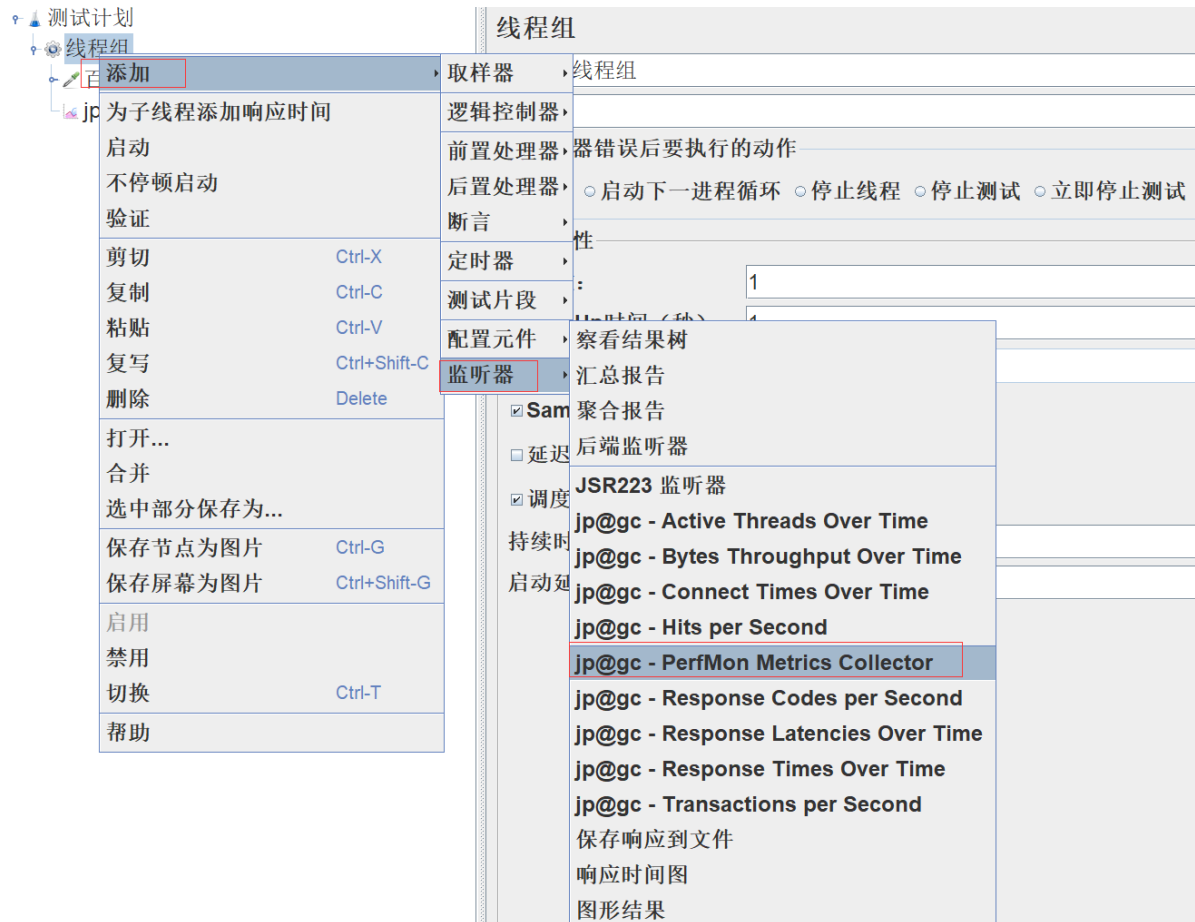
#### 1、解压 -- 运行



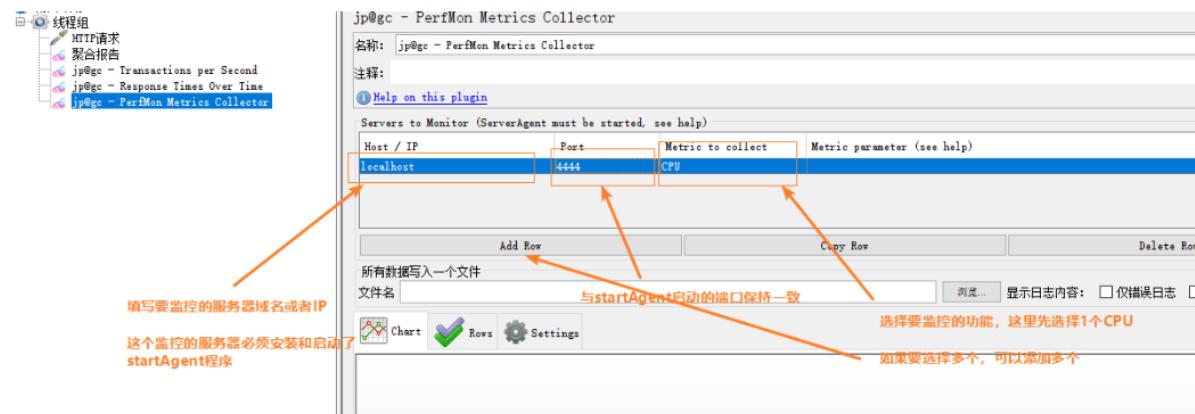
## 2、启动窗口

```
INFO 2021-06-07 15:00:07.725 [kg.apc.p] () : Binding UDP to 4444
INFO 2021-06-07 15:00:08.724 [kg.apc.p] () : Binding TCP to 4444
INFO 2021-06-07 15:00:08.726 [kg.apc.p] () : JP@GC Agent v2.2.3 started
```

## 3、jmeter当中，添加监控服务器资源使用率的组件

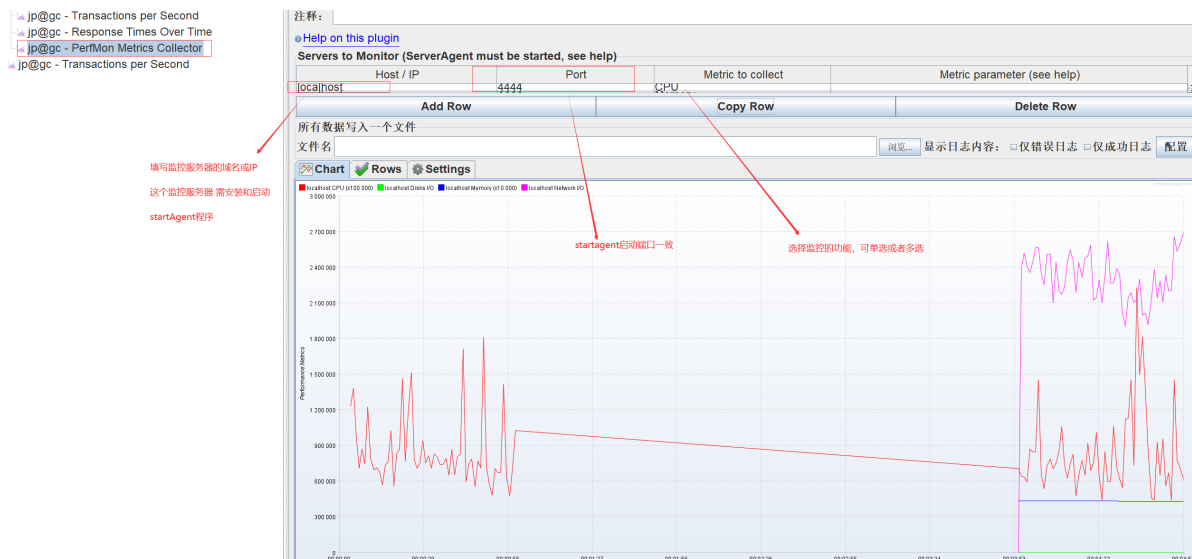


## 监控CPU



监控分别是,CPU，内存，网络和磁盘IO的监控点，查看结果





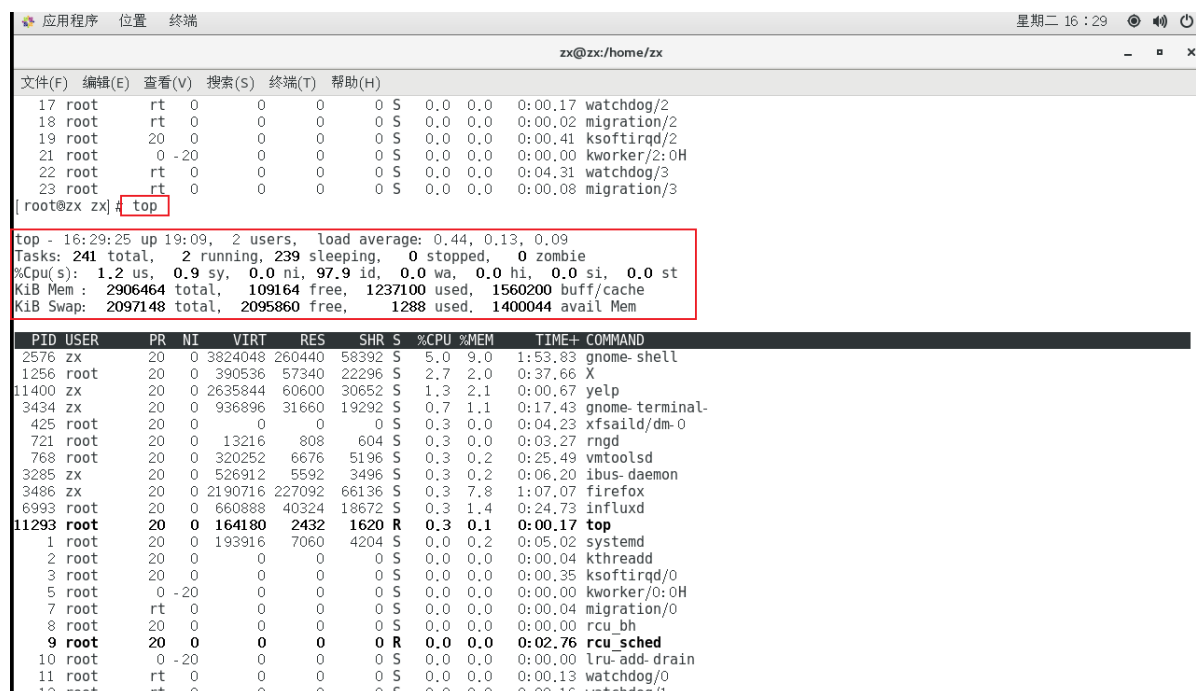
## 4 系统资源指标 (Linux)

### 4.1 Top命令

系统资源管理器

top命令类似与windows的任务管理器，查看内存、cpu、进程等操作信息

在Linux系统中常用top命令做资源性能分析工具



参数

1). 第一行：显示的系统时间，连接系统的用户和平均负载

load average: 0.44, 0.13, 0.09

系统当前时间

系统运行时间

users: 当前登录用户数

load average: 系统负载，即任务队列的平均长度-（1分钟、5分钟、15分钟）到现在的平均长度

## 2). 第二行 进程队列信息

```
Tasks: 241 total, 2 running, 239 sleeping, 0 stopped, 0 zombie
```

- Tasks : 201 total 进程总数

running 正在运行进程数

sleeping 睡眠进程数

## 3). 第三行 CPU信息

```
%Cpu(s): 1.2 us, 0.9 sy, 0.0 ni, 97.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
```

us:用户空间占用CPU百分比

-sy: 内核空间占用CPU百分比

- id: 空闲CPU百分比

## 4). 第四行 内存信息

```
KiB Mem : 2906464 total, 109164 free, 1237100 used, 1560200 buff/cache
```

- Mem : ktotal 物理内存总量

used 使用的物理内存总量

free 空闲内存总量 【关注】

buffers 用作内核缓存的内存量

## 5). 第五行 交换区内存

```
KiB Swap: 2097148 total, 2095860 free, 1288 used, 1400044 avail Mem
```

**Swap** :2097148 k **total** 交换分区总量

**1288 k used** 使用的交换区总量

**free** 空闲交换区总量

**cached** 缓冲的交换区总量

## 各进程（任务）的状态监控

**PID**: 进程ID, 进程的唯一标识符

**USER**: 进程所有者的实际用户名。

**PR**: 进程的调度优先级。这个字段的一些值是 '**rt**'。这意味这这些进程运行在实时态。

**NI**: 进程的**nice**值（优先级）。越小的值意味着越高的优先级。负值表示高优先级，正值表示低优先级

**VIRT**: 进程使用的虚拟内存。进程使用的虚拟内存总量，单位**kb**。**VIRT=SWAP+RES**

**RES**: 驻留内存大小。驻留内存是任务使用的非交换物理内存大小。进程使用的、未被换出的物理内存大小，单位**kb**。**RES=CODE+DATA**

**SHR**: **SHR**是进程使用的共享内存。共享内存大小，单位**kb**

**S**: 这个是进程的状态。它有以下不同的值：

- **D** - 不可中断的睡眠态。
- **R** - 运行态
- **S** - 睡眠态
- **T** - 被跟踪或已停止
- **Z** - 僵尸态

**%CPU**: 自从上一次更新时到现在任务所使用的**CPU**时间百分比。

**%MEM**: 进程使用的可用物理内存百分比。

**TIME+**: 任务启动后到现在所使用的全部**CPU**时间，精确到百分之一秒。

**COMMAND**: 运行进程所使用的命令。进程名称（命令名/命令行）

## 4.2 vmstat

查看内存明细

procs		-----memory-----				---swap--		-----io----		--system--		-----cpu-----				
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa	st
1	0	4152	141848	115956	376112	0	0	1	2	45	21	0	0	100	0	

**r**: 运行时的进程数量

**b:** 等待IO的进程数量

**swpd:** 交换的内存

**free:** 空闲的交换内存

**buff/cache:** 缓存的内存

**si:** 从交换内存进入到物理的内存数量

**so:** 从物理内存进入到交换内存的数量

**bi:** 写入磁盘块的数量

**bo:** 读取磁盘块的数量

**in:** 中断次数

**cs:** 上下文交换次数

## 4.3 iostat

### 查看io磁盘

说明: **iostat**是查看Linux系统io是否存在瓶颈很好用的一个命令;

语法: Usage: **iostat** [ options ] [ <interval> [ <count> ] ]

options:选项 interval:间隔 count:计数

```
[wind@localhost Desktop]$ iostat -x
Linux 2.6.32-754.el6.x86_64 (localhost.localdomain) 03/13/2020 _x86_64_ (1 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.08    0.00    0.24    0.01    0.00   99.67

Device:            rrqm/s   wrqm/s     r/s     w/s    rsec/s    wsec/s  avgrq-sz  avgqu-sz   await  r_awa
it w_await  svctm  %util
sda        0.01     0.35    0.05    0.21     2.00     4.43    25.21     0.00    2.49    0.
64    2.90    1.15    0.03
```

常用: **iostat -x 1 1**

(x:输出列, 1: 间隔1秒, 1: 采集1次)

CPU:

1.%user: 在用户级别运行所使用的CPU的百分比

2.%sys: 在系统级别(kernel)运行所使用CPU的百分比

3.%iowait: CPU等待硬件I/O时,所占CPU百分比

4.%idle: CPU空闲时间的百分比

Device: 【重点】

- 1.tps: 每秒钟发送到的I/O请求数
- 2.avgqu-sz: 是平均请求队列的长度,毫无疑问,队列长度越短越好
- 3.await: 每一个IO请求的处理的平均时间(单位是毫秒)
- 4.rkB/s: 每秒读取数据量(单位kb)
- 5.wkB/s: 每秒写入数据量(单位kb)
- 6.%util: 磁盘的繁忙程度,如接近100%那说明磁盘已经到瓶颈

## 4.4 free

查看内存

说明: 显示当前系统未使用的和已使用的内存数目,还可以显示被内核使用的内存缓冲区。

语法: free [options]

常用: free -m

(-m: 以MB为单位显示内存使用情况)

```
[wind@localhost Desktop]$ free -m
```

	total	used	free	shared	buffers	cached
Mem:	979	841	138	4	113	367
-/+ buffers/cache:		360	619			
Swap:	1983	4	1979			

通过这个命令要能够得出可用内存的大小:

可用内存=free + buffers + cached (参考值)

可用内存=138 + 113 + 367 = 618M

## 4.5 sar

查看网络

说明: sar命令可以通过参数单独查看系统某个局部的使用情况

语法: sar [options] [-A] [-o file] t [n]

命令: sar -n DEV 1 2

(-n: 网络设备;DEV:磁盘设备)

1). 1: 表示一秒采集一次信息,可自行设定

2). 2: 表示采集的次数，可自行设定

查看网络的使用情况，sar -n DEV 1 2 中“1”的含义是每1秒刷新1次。“2”代表总共获取两次数据

	IFACE	rxpck/s	txpck/s	rxkB/s	txkB/s	rxcmp/s	txcmp/s	rxmcs/s
11:20:01 AM	lo	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11:30:01 AM	eth0	0.17	0.00	0.01	0.00	0.00	0.00	0.00
11:30:01 AM	pan0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11:40:01 AM	lo	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11:40:01 AM	eth0	0.15	0.00	0.01	0.00	0.00	0.00	0.00
11:40:01 AM	pan0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11:50:01 AM	lo	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11:50:01 AM	eth0	0.18	0.02	0.01	0.00	0.00	0.00	0.00
11:50:01 AM	pan0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12:00:01 AM	lo	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12:00:01 AM	eth0	0.17	0.00	0.01	0.00	0.00	0.00	0.00
12:00:01 AM	pan0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average:	lo	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average:	eth0	0.81	0.35	0.92	0.02	0.00	0.00	0.00
Average:	pan0	0.00	0.00	0.00	0.00	0.00	0.00	0.00

IFACE: 代表是某个网口

rxpck/s: 每秒接收的数据包大小

txpck/s: 每秒发送的数据包大小

rxkB/s: 每秒接收的数据量大小（以KB为单位）

txkB/s: 每秒发送的数据量大小（以KB为单位）

txcmp/s: 每秒发送的压缩数据包

rxmcs/s: 每秒接收的多播数据包

## 4.6 nmon监控服务器资源的工具

说明：Nmon 是一个分析aix和linux性能的免费工具（其主要是ibm为自己的aix操作系统开发的，但是也可以应用在linux操作系统上

1. 解压文件
2. 文件重命名
3. 执行工具
4. 使用Excel分析工具分析

解压nmon的压缩包，然后选择对应操作系统的nmon工具来运行

```
tar -zxvf nmon_linux_14i.tar.gz
```

将文件重命名

```

-rw-r--r--. 1 root root 4487558 6月 8 19:53 nmon_linux_14i.tar.gz
-rwxrwxrwx. 1 root root 17228 10月 4 2013 nmonmerge_x86_64_debian6
-rwxrwxrwx. 1 root root 22130 9月 24 2013 nmonmerge_x86_64_sles11
-rwxrwxrwx. 1 root root 13556 10月 4 2013 nmonmerge_x86_debian6
-rwxrwxrwx. 1 root root 14019 10月 11 2013 nmonmerge_x86_GNU_2.0.0
-rwxrwxrwx. 1 root root 17503 10月 11 2013 nmonmerge_x86_GNU_2.6.15
-rwxrwxrwx. 1 root root 18768 10月 11 2013 nmonmerge_x86_GNU_2.6.24
-rwxrwxrwx. 1 root root 14019 10月 2 2013 nmonmerge_x86_puppy431
-rwxrwxrwx. 1 root root 18968 9月 24 2013 nmonmerge_x86_sles11
-rwxrwxrwx. 1 root root 222076 9月 17 2013 nmon_x86_64_centos6
-rwxrwxrwx. 1 root root 228085 10月 4 2013 nmon_x86_64_debian5
-rwxrwxrwx. 1 root root 228085 9月 24 2013 nmon_x86_64_debian6
-rwxrwxrwx. 1 root root 293513 9月 17 2013 nmon_x86_64_debian7
-rwxrwxrwx. 1 root root 241978 10月 2 2013 nmon_x86_64_fatdog64_601
-rwxrwxrwx. 1 root root 281888 10月 3 2013 nmon_x86_64_fedora17
-rwxrwxrwx. 1 root root 288656 9月 18 2013 nmon_x86_64_fedora18

```

## 使用：

输入命令:nmon -s 3 -c 10 -f -m 目录名称（目录必须存在）

-s：收集数据的频率

-c：收集数据的次数

-f：生成nmon数据文件

-m：指定数据文件的目录名称

```

[root@zx nmon]# ./nmon -s3 -c10 -f -m /home/zx/桌面
[root@zx nmon]#

```

路径

运行后，带有桌面的系统，在当前的Desktop中能看到这个文件



将.nmon后缀的文件放在windows系统中

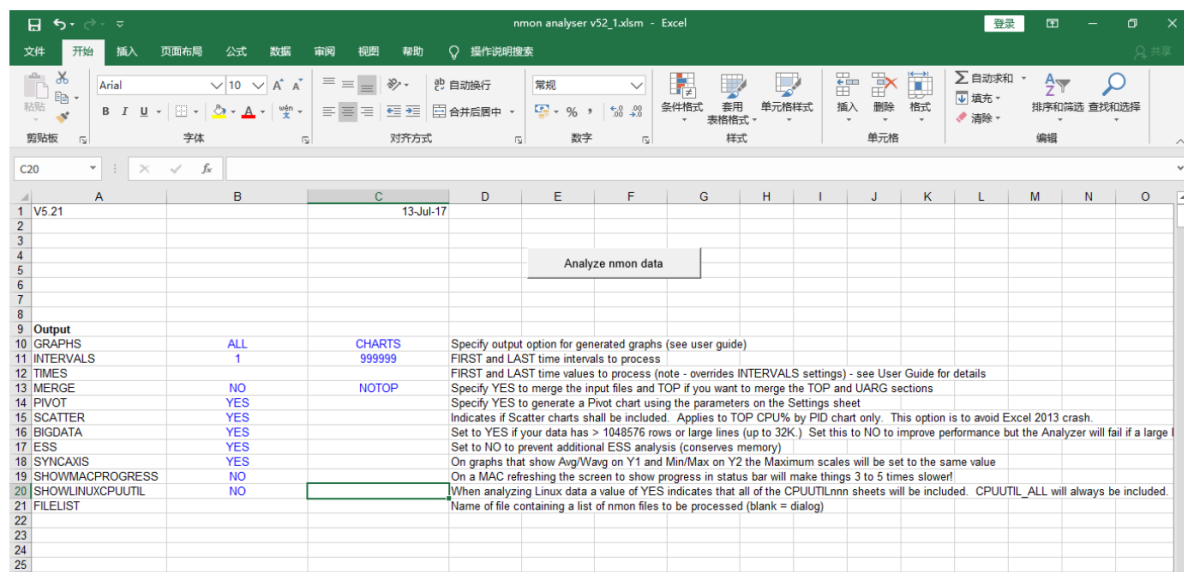
名称	类型	大小
nmon_linux_14i.tar.gz	360压缩	4,383 KB
nmon使用方式.txt	文本文档	1 KB
zx_210608_1955.nmon	NMON 文件	44 KB
zx_210608_1955.nmon.xlsx	Microsoft Excel ...	142 KB

使用Nmon\_analyser工具分析

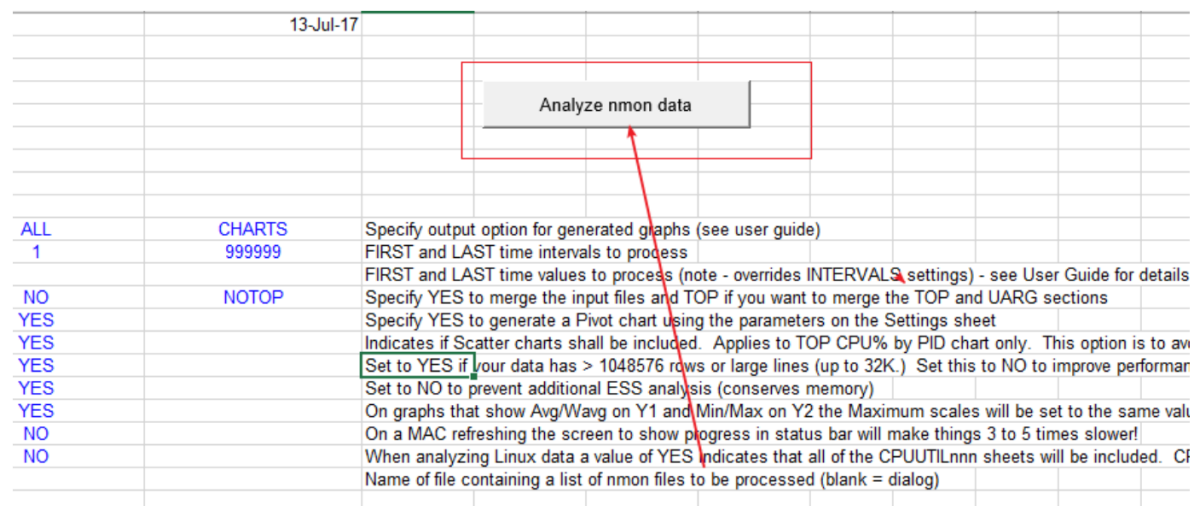
打开nmon\_analyser

名称	类型	大小
nmon analyser v52_1.xlsm	Microsoft Excel ...	217 KB

打开效果：



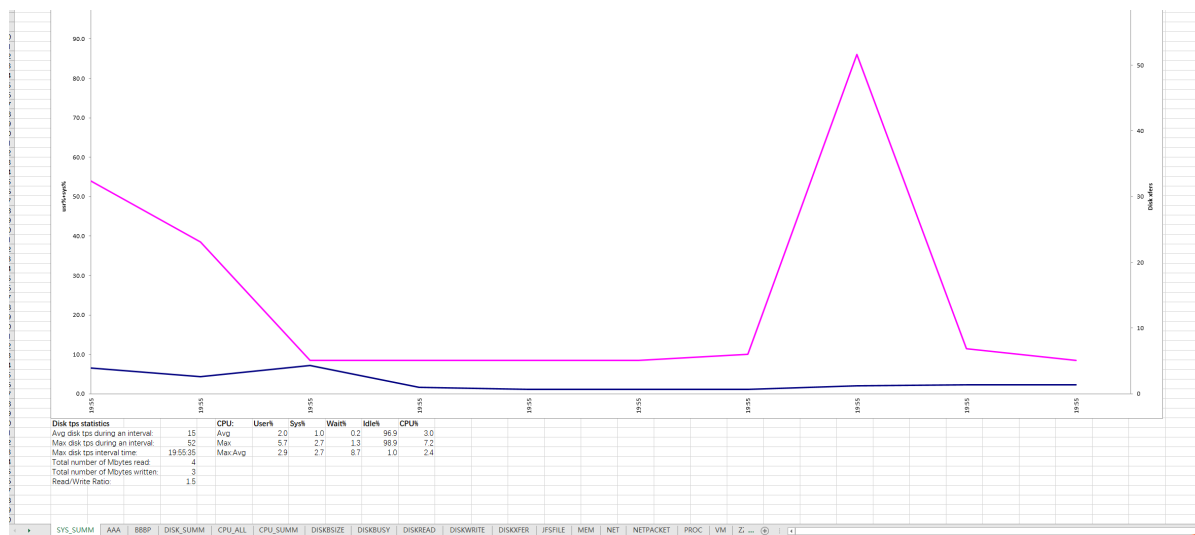
然后点击Analyze nmon data



有一个加载过程显示：

就可以打开文件了





使用nmon工具，可以用来帮助我们整体性的分析服务端的CPU，内存，网络，IO，虚拟内存等指标。比使用jmeter客户端的PerfMon Metris Collector组件更好用，也更灵活。

区别是，nmon是离线采集，而PerfMon Metris Collector是实时采集。

应用场景：下班前，开启nmon采集，然后开启压测，第二天打开采集的数据，分析一晚上的压测成果。

## 5 TPS计算公式

需求：如何确定测试环境的TPS指标



PV: (Page View) 即页面访问量，每打开一次页面PV计数+1，刷新页面也是。PV只统计页面访问次数。

UV(Unique Visitor),唯一访问用户数，用来衡量真实访问网站的用户数量。

一般用UV统计用户活跃数，用PV统计用户访问页面的频率

## 1.1 普通计算方法

计算公式：TPS= 总请求数 / 总时间

不准确，比如一天的访问量存在峰值，是有很大的波动的。所以就不能单纯的去这样计算

按照需求所示，在2019年第32周，日均有4.13万的浏览量，那么总请求数，我们可以认为估算为4.13万（1次浏览都至少对应1个请求）  
总请求数 = 4.13 万请求数 = 41300 请求数

总时间：由于不知道每个请求的具体时间，我们按照普通方法，我们可以按照一周的时间进行计算  
总时间 = 1 天 = 1 \* 24 小时 = 24 \* 3600 秒

tps = 41300请求书 / 24\*3600s = 0.48请求/s

## 1.2 二八原则计算方法

二八原则就是指80%的请求在20%的时间内完成

计算公式：TPS = 总请求数 80% / (总时间20%)

按照公式进行计算

TPS = 41300 \* 0.8请求数 / 24\*3600\*0.2秒 = 1.91 请求数/秒

结论：按照二八原则计算，在测试环境我们的TPS只要能达到1.91请求数每秒就能满足线上需要。二八原则的估算结果会比平均值的计算方法更能满足用户需求

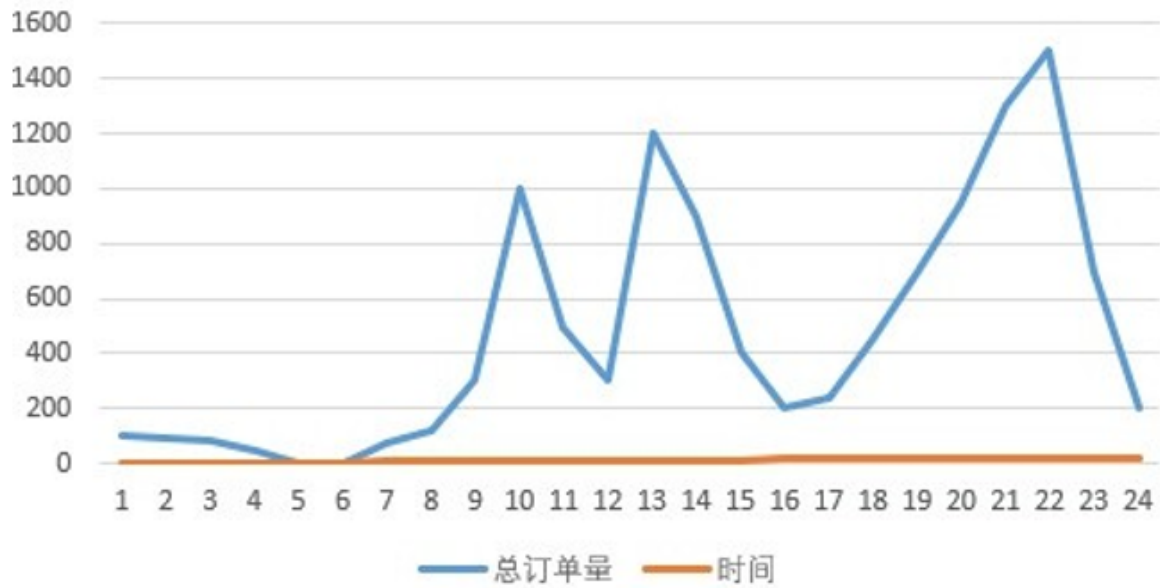
不论是普通计算方法还是二八原则计算方法，都是估算的方法，不够准确。

## 1.3 按照业务数据进行计算

业务数据：有的公司会统计一定时间内的所有业务数据，我们可以根据这个业务数据曲线图，统计出最多请求的数量和时间比例。

曲线图看趋势

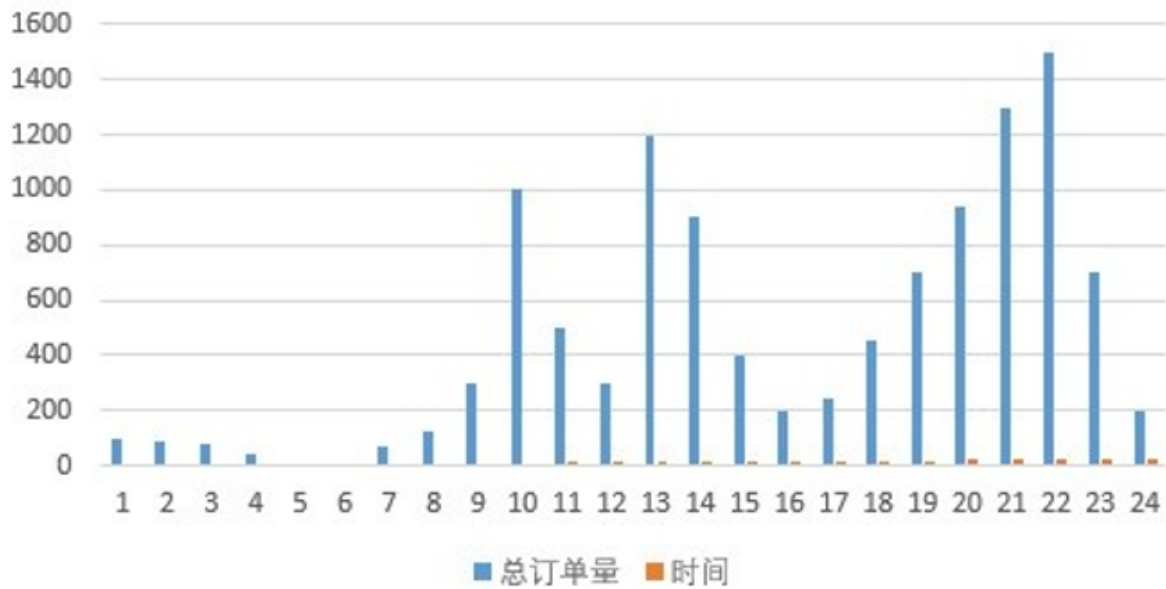
图表标题



总数：40474 个

直方图看数据和趋势

图表标题



计算模拟用户正常业务操作（稳定性测试）的并发量：

根据这些数据统计图，可以得出结论：

$$\text{TPS} = 40474 * 0.8 / 16 * 3600 * 0.2 = 2.81 \text{ 请求/s}$$

计算模拟用户峰值业务操作（压力测试）的并发量：

根据这些数据统计图，可以得出结论：

21点 —— 22点 8853个

$TPS = 8853 / 3600S = 2.22$  请求/s

**tps系数**

2-8倍

$TPS = 8853 * 8 / 3600 = 17.76$  请求/s

## 6 线程组（特殊线程组）

---

**setUp线程组**

一种特殊类型的线程组，用于在执行常规线程组之前执行一些必要的操作。在“setup thread group”下提到的线程行为与普通线程组完全相同。不同的是执行顺序 --- **它会在普通线程组执行之前被触发。**

应用场景举例：

- A、测试数据库操作功能时，用于执行打开数据库连接的操作。
- B、测试用户购物功能时，用于执行用户的注册、登录等操作。

**tearDown线程组**

一种特殊类型的ThreadGroup，用于在执行常规线程组完成后执行一些必要的操作。在“teardown thread group”下提到的线程行为与普通线程组完全相同。不同的是执行顺序---**它会在普通线程组执行之后被触发。**

应用场景举例：

- A、测试数据库操作功能时，用于执行关闭数据库连接的操作。
- B、测试用户购物功能时，用于执行用户的退出等操作。