JMeter 是一个流行的用于**[负载测试](http://www.ltesting.net/html/95/category-catid-95.html" \t "_blank)**的[**开源**](http://www.ltesting.net/html/3/category-catid-3.html)工具， 具有许多有用的功能元件，如线程组（thread group）， 定时器（timer）， 和HTTP 取样 (sampler) 元件。 本文是对JMeter 用户手册的补充，而且提供了关于使用Jmeter的一些模拟元件**[开发](http://www.ltesting.net/html/4/category-catid-4.html" \t "_blank)**[**质量**](http://www.ltesting.net/html/5/category-catid-5.html)[**测试脚本**](http://www.ltesting.net/html/96/category-catid-96.html)的指导。

|  |
| --- |
|  |

　　本文同时也讨论了一项重要的内容：在指定了精确的响应时间要求后，如何来校验测试结果，特别是在采用了置信区间分析这种严格的统计方式的情况下应如何操作。请注意，我假定本文的读者们了解关于Jmeter的基础**[知识](http://www.ltesting.net/ask/" \t "_blank)**，本文的例子基于Jmeter2。0。3版。   
  
**确定一个线程组的ramp-up period （Determine）**  
　　Jmeter脚本的第一个要素是线程组（Thread Group），因此首先让我们来回顾一下。 正如图一所示，线程组需要设置以下参数：   
　　·线程数量。   
　　·ramp-up period。   
　　·运行测试的次数。   
　　·启动时间:立即或者预定的时间，如果是后者，线程组所包含的元素也要指定这个起止时间。

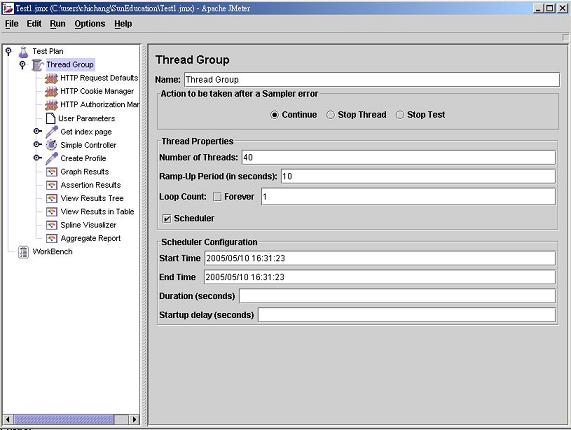
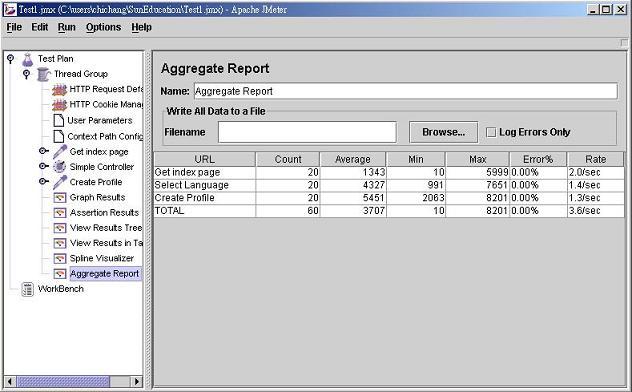


　　图 1。 JMeter 线程组（JMeter Thread Group）  
  
　　每个线程均独立运行**[测试计划](http://www.ltesting.net/html/74/category-catid-474.html" \t "_blank)**。因此， 线程组常用来模拟并发用户访问。如果客户机没有足够的能力来模拟较重的负载，可以使用Jmeter的分布式测试功能来通过一个Jmeter控制台来远程控制多个Jmeter引擎完成测试。   
  
　 　参数 ramp-up period 用于告知JMeter 要在多长时间内建立全部的线程。默认值是0。如果未指定ramp-up period ，也就是说ramp-up period 为零， JMeter 将立即建立所有线程，假设ramp-up period 设置成T 秒， 全部线程数设置成N个， JMeter 将每隔T/N秒建立一个线程。   
  
　　线程组的大部分参数是不言自明的，只有ramp-up period有些难以理解， 因为如何设置适当的值并不容易。 首先，如果要使用大量线程的话，ramp-up period 一般不要设置成零。 因为如果设置成零，Jmeter将会在测试的开始就建立全部线程并立即发送访问请求， 这样一来就很容易使[**服务器**](http://www.ltesting.net/html/78/category-catid-378.html)饱和，更重要的是会隐性地增加了负载，这就意味着服务器将可能过载，不是因为平均访问率高而是因为所有线程的第一次并发访问而引起的不正常的初始访问峰值，可以通过Jmeter的聚合报告监听器看到这种现象。

这种异常不是我们需要的，因此，确定一个合理的ramp-up period 的规则就是让初始点击率接近平均点击率。当然，也许需要运行一些测试来确定合理访问量。   
  
　　基于同样的原因，过大的ramp-up period 也是不恰当的，因为将会降低访问峰值的负载，换句话说，在一些线程还未启动时，初期启动的部分线程可能已经结束了。   
  
　 　那么，如何检验ramp-up period I太小了或者太大了呢？首先，推测一下平均点击率并用总线程除点击率来计算初始的ramp-up period。 例如，假设线程数为100， 估计的点击率为每秒10次， 那么估计的理想ramp-up period 就是 100/10 = 10 秒。 那么，应怎样来提出一个合理的[**估算**](http://www.ltesting.net/html/65/category-catid-165.html)点击率呢？没有什么好办法，必须通过运行一次测试脚本来获得。  
  
　 　其次， 在测试计划（test plan）中增加一个聚合报告监听器，如图2所示，其中包含了所有独立的访问请求（一个samplers）的平均点击率。 第一次取样的点击率（如http请求）与ramp-up period 和线程数量密切相关。通过调整ramp-up period 可以使首次取样的奠基率接近平均取样的点击率。

[](http://www.ltesting.net/uploads/2007/06/1_200706212342472.jpg)

[点击查看大图](http://www.ltesting.net/uploads/2007/06/1_200706212342472.jpg)

　　图2 JMeter 聚合报告   
  
　　第三， 查验一下Jmeter日志(文件位置：JMeter\_Home\_Directory/bin) 的最后一个线程开始时第一个线程是否真正结束了，二者的时间差是否正常。  
  
　　总之，是否能确定一个适当的ramp-up time 取决于以下两条规则:   
　　·第一个取样器的点击率（hit rate）是否接近其他取样器的平均值，从而能否避免ramp-up period 过小。  
　　·在最后一个线程启动时，第一个线程是否在真正结束了，最好二者的时间要尽可能的长，以避免ramp-up period过大。  
  
　 　有时，这两条规则的结论会互相冲突。 这就意味着无法找到同时满足两条规则的合适的ramp-up period。 糟糕的测试计划通常会导致这些问题，这是因为在这样的测试计划里，取样器将不能充分地采集数据，可能因为测试计划执行时间太短并且线程会很快的运行结束。   
  
  
  
**用户思考时间（User think time），定时器，和代理服务器（proxy server）**  
　 　在负载测试中需要考虑的的一个重要要素是思考时间（think time）， 也就是在两次成功的访问请求之间的暂停时间。 有多种情形挥发导致延迟的发生： 用户需要时间阅读文字内容，或者填表， 或者查找正确的链接等。未认真考虑思考时间经常会导致测试结果的失真。例如，估计数值不恰当，也就是被测系统可以支持的最多用户量（并发用户）看起来好像 要少一些等。

　　Jmeter提供了一整套的计时器（timer）来模拟思考时间（think time）， 但是仍旧存在一个问题：: 如何确定适当的思考时间呢？幸运的是， JMeter 提供了一个不错的答案：使用 JMeter HTTP 代理服务器（Proxy Server）元件。   
  
　　代理服务器会记录在使用一个普通的浏览器（如FireFox 或 Internet Explorer）浏览一个web应用时的操作。 另外， JMeter 在记录操作的同时会建立一个测试计划（test plan）。 这个功能能提供以下便利：   
  
　　·不必手工建立HTTP 访问请求， 尤其是当要设置一些令人乏味的参数时（然而，非英文的参数也许不能正常工作) 。JMeter 将会录制包括隐含字段（hidden fields）在内的所有内容。   
  
　　·在生成的测试计划中，Jmeter会包含浏览器生成的所有的 HTTP 报头，如User-Agent (e。g。， Mozilla/4。0)， 或A[**clearcase**](http://www.ltesting.net/ceshi/ceshijishu/rjcsgj/rational/%3CSTRONG%3E%3CA%20href=)/" target="\_blank" >cceptLanguage (e。g。， zh-tw，en-us;q=0。7，zh-cn;q=0。3)等。   
  
　　·JMeter 会根据设置在录制操作的同时建立一些定时器，其延迟时间是完全根据真实的操作来设置的  
现 在让我们来看一下如何配置Jmeter的录制功能。 在JMeter 的控制台上， 在工作台（WorkBench）元件上单击右键，然后选择”add the HTTP Proxy Server “。 注意是在WorkBench 上单击右键而不是在Test Plan上， 因为现在是要为记录操作进行配置而不是要运行测试计划。  HTTP Proxy Server 的实现原理就是通过配置浏览器的代理服务器而使所有的访问请求通过JMeter发送（，因而被Jmeter把访问过程录制下来）。   
  
　　如图3所示， HTTP代理服务器（HTTP Proxy Server）元件的一些参数必须被配置:   
  
　　·端口(port): 代理服务器的监听端口   
  
　　·目标控制器（Target Controller）: 是代理用于存储生成的数据的控制器，默认情况下，， JMeter 将会在当前的测试计划中找一个记录用的控制器用于存储，此外也可以在下拉菜单中选择任意控制起来存储，通常默认值就可以了。   
  
　 　·分组(Grouping): 确定在测试计划中如何来为生成的元件分组。 有多个选项， 一般可以选择“只存储每个组的第一个样本”，否则，将会原样录制URLs，包括包含图像和JavaScripts脚本的页面。当然 也可以尝试一下默认值“不对样本分组”（"Do not group samples"），来看一下JMeter 建立的原版的测试计划。   
  
　　·包含模式(Patterns to Include) 和 排除模式(Patterns to Exclude) :帮助过滤一些不需要的访问请求。

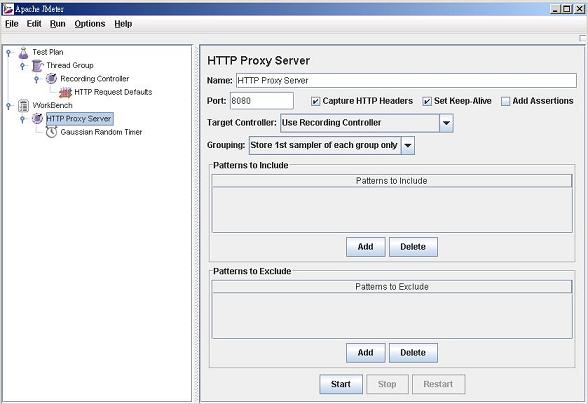
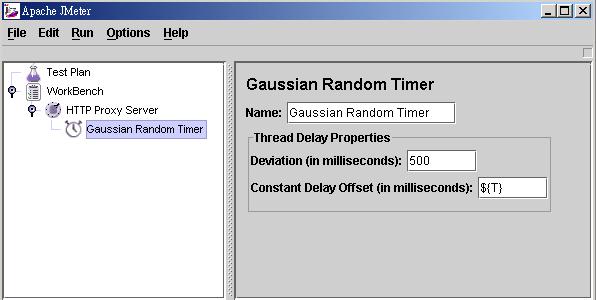


　　图 3。 JMeter 代理服务器(Proxy Server)。  
  
　　当你点击开始 （Start）按钮时，代理服务器就会开始记录所接受的HTTP 访问请求。 当然，在开始记录前，要首先设置好浏览器的代理服务器设置。在代理服务器元件中可以增加一个定时器子元件（配置元件），用于告知Jmeter来在其生成的 HTTP请求中自动的增加一个定时器。Jmeter会自动把实际的延迟时间存储为一个被命名为T的Jmeter变量，因此，如果在代理服务器元件里使用了 高斯随机定时器，就应该在其中的固定延迟偏移（Const[**ant**](http://www.ltesting.net/html/57/category-catid-457.html) Delay Offset）设置项里添上${T}（用于自动引用纪录的延迟时间），如图4所示。这是另一个节省时间的便利特性。

[](http://www.ltesting.net/uploads/2007/06/1_200706212342474.jpg)

[点击查看大图](http://www.ltesting.net/uploads/2007/06/1_200706212342474.jpg)

　　图 4。 在代理服务器组建中增加一个高斯随机定时器   
  
  
　　定时器将会使相应的的取样器被延迟。 延时的规则是，在上一个访问请求被响应并延时了指定的时间后，下一个被定时器影响的取样访问请求才会被发送出去。 因此， 你必须手工删除第一个取样器中自动生成的定时器，因为第一个取样器不需要定时器。   
  
　 　在启动HTTP代理服务器以前，要在测试计划中增加一个线程组（thread group），在线程组中增加一个录制控制器（recording controller）用于存储生成的结果。 否则， 生成的元件将会被直接添加到工作台里。另外， 在录制控制器里增加一个HTTP请求默认值元件HTTP Request Defaults 元件 (是一个配置元件) 也很重要，这样Jmeter就不填写使用了默认值的字段。   
  
　　录制完成后， 停止HTTP 代理服务器; 在录制控制器元件上单击右键将记录的元件保存为一个文件用于以后重用，另外，不要忘了恢复浏览器的代理服务器设置。   
  
**指定响应时间**[**需求**](http://www.ltesting.net/html/62/category-catid-162.html)**并校验结果**

    　尽管本节内容与Jmeter不是直接相关，但是Jmeter仍旧是指定响应时间需求和校验测试结果这两个负载测试评价任务互相联系的纽带。  
  
　 　在web应用的环境里，响应时间指的是从提交访问请求到等到HTML结果所耗费的时间。从技术的角度看，响应时间也应包括浏览器重绘HTML页面的时 间，但是浏览器一般是一块接着一块地显示而不是直接显示完整的整个页面，让人感觉响应时间要少一些。 另外，典型的情况是，负载**[测试工具](http://www.ltesting.net/html/2/category-catid-2.html" \t "_blank)**不会考虑浏览器的重绘时间。 因此， 在实际的[**性能测试**](http://www.ltesting.net/html/95/category-catid-95.html)中，我们将考虑以上描述的情形， 如果不能确信，可以在正常的响应时间上加一个固定值，如0.5秒。  
  
　　以下是一套众所周知的确定相应时间的标准:   
　　·用户将不会注意到少于0.1秒的延迟   
　　·少于1秒的延迟不会中断用户的正常思维， 但是一些延迟会被用户注意到   
　　·延迟时间少于10秒，用户会继续等待响应   
　　·延迟时间超过10秒后，用户将会放弃并开始其他操作   
  
　 　这些阀值很有名并且一般不会改变，因为是关乎人类的感知特性的。 所以要根据这些规则来设置响应时间需求， 也需要适当调整以适应实际应用。例如，亚马逊公司(Amazon.com) 的主页也遵循了以上规则，但是由于更偏重于风格上的一致，所以在响应时间上有一点损失。   
  
　　乍一看，好像有两种不同的方式来确定相应时间需求：   
　　·平均响应时间(Average response time )  
　　·绝对响应时间（Absolute response time）;即， 所有的响应时间必须低于某一阀值  
  
　　指定平均响应时间比较简单一些（straightforward），但是由于数据变化的干扰，这个需求往往难以实现。为什么取样中的20%的响应时间要比平均值高3倍以上呢？请注意，JMeter 计算平均响应时间与图形结果监视器中的标准偏差是一致的。   
  
　　另一方面， 对绝对响应时间需求过于苛求是不实际的。 如果只有0。5%的取样不能通过测试该怎么办？如果再测一次，又会有很大的变化。 幸运的是， 使用置信区间（confidence interva）分析这种正规的统计方法可以顾及到取样变化的影响。   
在继续进行前，让我们首先回顾一些基本的统计学知识。   
  
**中心极限定理（The central limit theorem）**   
   
　　中心极限定理表明如果总体的分布有一个平均值μ和标准偏差σ，那么对于一个十分大的n(>30)，其取样平均值的分布将接近于正态分布，其平均值μmean = μ ，标准偏差σmean = σ/√n。   
注意取样平均值的分布是正态的，而取样自身的分布不必是正态的。也就是说如果多次运行测试脚本则测试结果的平均响应时间将会是正态的。   
  
　 　图 5 和图 6 分别展示了两个正态分布。 在这里横坐标是采样响应时间的均值， 总体的均值被调整到坐标的原点（shifted so the population mean is at the origin）。 图5 表明90%的时间里，采样均值位于±Zσ的区间里（percent of the time， the sampling means are within the interval ±Zσ，），这里的Z=1.645 和 σ 是标准偏差。 图 6 表明了99%的情况下的情形这时的Z=2.576。 在给定的概率下，如90%， 我们可以看到相应的Z呈现正态曲线，反之亦然。

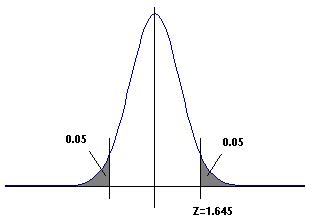


　　Figure 5。 Z value for 90 percent

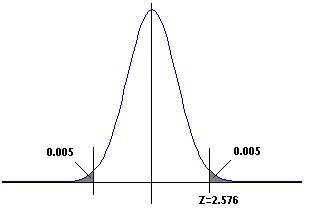


　　Figure 6。 Z value for 99 percent  
  
　　在相关资料中所列 的是可提供正态曲线计算的一些网站。在这些网站，我们可以计算随意的相对区间内的概率(如，-1.5 < X < 1.5)或者在一个聚集的区域（cumulated area）内 ，(如， X < 1.5)。 也可以从下面的表中得到近似值。   
  
　　表 1。 对应于给定的置信区间（confidence interval）的标准偏差范围（Standard deviation range）



　　表 2。 对应于给定的标准偏差范围（Standard deviation）的置信区间（confidence interval）



**置信区间（Confidence interval）**  
   
　　置信区间 （confidence interval）的定义是[取样平均值- Z\*σ/√n， 取样平均值+ Z\*σ/√n]。 例如， 如果置信区间（概率）是90%， 经查找可知Z 值是1。645， 于是置信区间就是 [取样平均值- 1。645\*σ/√n， 取样平均值+ 1。645\*σ/√n]， 这意味着在90%的时间里， 总体平均值（population mean）(是未知的) 会落入这个置信区间内。 也就是说， 我们的测试结果是十分接近的。 如果 σ（标准偏差） 更大一些， 置信区间也会更大，这就意味着置信区间的上限就会更可能会越过可以接受的范围，即σ 越大，结果越不可信。   
  
**响应时间需求（Response-time requirements ）**  
      
　　现在我们把所有的信息都归结到响应时间需求上来。首先。必须要定义性能需求，如: %95概率的置信区间的平均响应时间的上限必须小于5秒。 当然，最好有相应的需求或场景。   
  
　 　在性能测试结束后，假设进分析得出结论是平均响应时间是4.5秒，标准偏差时4.9秒，样本数量是120个，然后就可以计算%95概率的置信区间了。 通过查表1，找到Z值是 1。95996。 于是置信区间就是 [4.5 – 1.95996\*4.9/√120， 4.5 + 1.95996\*4.9/√120]， 也就是 [3.62， 5.38]。 尽管看起来这个响应时间看起来很不错，但这个结果（因为超出了需求的要求，因而）是不可接受的。 实际上， 可以检验的是即使是对于80%概率的可信区间，这个测试结果也是不能接受的。正如你所看到的，使用了置信区间分析后，会得到一个十分精确的方法来估算测试 质量。

　　在web应用中，为了测定某一场景的响应时间，我们一般要通过测试工具来发送多个访问请求，例如：   
　　4. 登陆   
　　5. 显示表单   
　　6. 提交表单  
  
　　假设我们对请求3更感兴趣。为进行置信区间分析，我们需要的仅是请求3的所有样本的响应时间均值和标准偏差，而不是全部被统计的样本的。   
  
　　在Jmeter的图表结果监听器中计算的却是全部请求的响应时间均值和标准偏差。 而Jmeter的聚合报告监听器计算的是独立的采样器的响应时间均值，可惜没有计算标准偏差。   
总之， 仅仅指定响应时间均值是危险的， 因为不能反映出数据的变化。 即使响应时间均值是可以接受的，但是置信区间仅有75％，这个结果也不能令人信服。但是，使用置信区间分析还是会带来更多的确定性。   
  
**结论**  
  
　　本文讨论了以下内容:   
　　·详细讲解了Jmeter 线程组在加载负载时的特别设置   
  
　　·使用Jmeter代理服务器(Proxy Server)元件自动建立测试脚本的指导方针，其重点在于模拟用户思考时间（user think time ）。  
  
　　·置信区间分析（Confidence interval analysis）， 一种我们可以用来更好地满足响应时间需求的统计分析方法   
  
　　通过使用本文提及的技术可以改善测试脚本的质量，更广泛地说，本文所讨论的内容属于是性能测试的一个工作流程的一部分， 是其中的一个较困难的部分。性能测试包括并不仅限于以下内容：   
  
　　·编写性能测试需求  
　　·选择测试情景   
　　·准备**[测试环境](http://www.ltesting.net/html/74/category-catid-374.html" \t "_blank)**  
　　·编写测试脚本   
　　·执行测试  
　　·回顾测试脚本和测试结果   
　　·指出性能瓶颈   
　　·书写测试报告  
  
　　此外， 性能测试结果，包括确定下来的瓶颈， 都需要反馈给开发团队或者架构师进行优化设计。 在这个过程中，并写测试脚本和回顾测试脚本是其中很重要的部分，要精心筹划和管理实施。凭借测试脚本指导和一个好的性能[**测试流程**](http://www.ltesting.net/html/97/category-catid-97.html)，你将会有更多的机会来在较重负载下优化软件性能。  
  
  
**关于作者**  
  
　　Chi-Chang Kung 是台湾Sun 公司的java系统架构师，也是IEEE 和ACM的成员。   
  
**相关资源**   
  
　　·JMeter: http://jakarta.apache.org/jmeter/index.html   
　　·《中心极限理论以及经典推论》（"Central Limit Theorem and Classical Inference" ）Scott M。 Lynch (2005年2月): [http://www.princeton.edu/~slynch/clt\_inference.pdf](http://www.princeton.edu/%7Eslynch/clt_inference.pdf)   
  
　　·置信区间（Confidence intervals）: http://people.hofstra.edu/faculty/Stefan\_Waner/RealWorld/finitetopic1/confint.html   
  
　　·《java网站的**[性能分析](http://www.ltesting.net/html/95/category-catid-95.html" \t "_blank)**》 （Performance Analysis for Java Websites）， Stacy Joines et al. (Addison-Wesley， 2002年9月; ISBN: 0201844540): http://www.amazon.com/exec/obidos/ASIN/0201844540/javaworld   
  
·《响应 时间：三个重要的限制条件》("Response Times: The Three Important Limits") 引自《实用工程学》( Usability Engineering)， Jakob Nielsen (Morgan Kaufmann， 1994; ISBN 0125184069): http://www.useit.com/papers/responsetime.html   
  
　　·一些提供了正态曲线计算功能的网站(Websites for normal curve calculation):   
　　o        http://www.psychstat.smsu.edu/introbook/normal.htm   
　　o        http://www.ecositebr.bio.br/curva\_normal.htm   
　　o        http://statistik.wu-wien.ac.at/mathstat/hatz/vo/applets/probCalc/normal\_z\_p.html  
  
　　·更多关于测试的文章，请参照JavaWorld's 标题索引的Testing 部分: http://www.javaworld.com/channel\_content/jw-testing-index.shtml   
  
　　·关于JAVA开发工具，参见JavaWorld's 标题索引的Development Tools 部分: http://www.javaworld.com/channel\_content/jw-tools-index.shtml

本文标题：[**JMeter技巧集锦**](http://www.ltesting.net/ceshi/open/kyxncsgj/jmeter/2007/0622/60945.html)  
<http://www.ltesting.net/ceshi/open/kyxncsgj/jmeter/2007/0622/60945.html>