实验报告四

|  |  |
| --- | --- |
| 姓名 | fyl |
| 学号 | SA18225079 |
| 工具 | Win10；Dev C++；  WAS（Microsoft的Web Application Stress Tool） |

目录

[1. 压力测试说明 3](#_Toc28046)

[1.1测试必要性 3](#_Toc3744)

[1.2 WAS概要介绍 3](#_Toc6038)

[1.3开始使用WAS 4](#_Toc14625)

[2. 录制测试脚本 5](#_Toc23656)

[2.1.浏览器准备 5](#_Toc25421)

[2.2记录脚本 7](#_Toc8994)

[2.2设置脚本参数 9](#_Toc12812)

[2.2.1 Content Tree 9](#_Toc30274)

[2.2.2 Setting（设置） 9](#_Toc564)

[2.2.3 Perf Counters（性能计数器） 10](#_Toc4690)

[2.2.4 Page Groups 11](#_Toc19720)

[2.2.5 Users 12](#_Toc1765)

[2.2.6 Clients 12](#_Toc22983)

[2.2.7 Cookies 12](#_Toc25700)

[3. 运行测试脚本 12](#_Toc27825)

[4. 运行结果 12](#_Toc1420)

[4.1 摘要 12](#_Toc26407)

[4.2 Result Codes 12](#_Toc14725)

[4.3 Pref Counters 13](#_Toc10702)

[4.4 Scripts Settings 13](#_Toc31734)

[4.5 Test Clients 13](#_Toc20711)

[4.6 Page Summary 13](#_Toc4746)

[4.7 Page Groups 13](#_Toc25649)

[4.8 Page Data 14](#_Toc25530)

[5. 其他检测 14](#_Toc7955)

[5.1 DOS检测 14](#_Toc14560)

[5.2 CPU检测 14](#_Toc23263)

[5.3 CPU使用率 15](#_Toc28123)

[6. 结论 15](#_Toc13424)

1. **压力测试说明**

## 1.1测试必要性

随着服务器端处理任务的日益复杂以及网站访问量的迅速增长，服务器性能的优化也成

了非常迫切的任务。在优化之前，最好能够测试一下不同条件下服务器的性能表现。找出性能瓶颈所在是设计性能改善方案之前的一个至关紧要的步骤。

负载测试是任何 Web 应用的开发周期中一个重要的步骤。如果你在构造一个为大量用

户服务的应用，搞清楚你的产品配置能够承受多大的负载非常重要。如果你在构造一个小型的 Intranet 网站，测试能够暴露出最终会导致服务器崩溃的内存漏洞以及竞争情况。

但是在实际的开发过程中，要按照实际投入运行的情况，组织成千上万的用户来进行压力测试，无论从那个方面看，都是不现实的。而且这样一旦发现了问题，不仅需要重复的进行这种耗费巨大的测试，而且问题不容易重现，不能方便的找出性能的瓶颈所在。而使用软件进行压力测试就不会存在这种情况。

无论是哪种情形，花些时间对应用进行负载测试可以获得重要的基准性能数据，为未来的代码优化、硬件配置以及系统软件升级带来方便。即使经费有限的开发组织也可以对它们的网站进行负载测试，因为 Microsoft 的压力测试工具 WAS 是可以免费下载的。

## 1.2 WAS概要介绍

**1．使用WAS的好处**

**简单**

WAS允许你以不同的方式创建测试脚本：你可以通过使用浏览器走一遍站点来录制脚本，可以从服务器的日志文件导入URL，或者从一个网络内容文件夹选择一个文件。当然，你也可以手工地输入URL来创建一个新的测试脚本。

不像其它的工具，你可以使用任何数量的客户端运行测试脚本，全部都有一个中央主客户端来控制。在每一个测试开始前，主客户机透明地执行以下任务：

|  |  |
| --- | --- |
| 标识 | 任务 |
| 1 | 与其他所有的客户机通讯 |
| 2 | 把测试数据分发给所有的客户端 |
| 3 | 在所有客户端同时初始化测试 |
| 4 | 从所有的客户端收集测试结果和报告 |

这个特性非常重要，尤其对于要测试一个需要使用很多客户端的服务器群的最大吞吐量时非常有用。

**高可用性**

WAS是被设计用于模拟Web浏览器发送请求到任何采用了HTTP1.0或1.1标准的服务器，而不考虑服务器运行的平台。

除了它的易用性外，WAS还有很多其它的有用的特性，包括：

|  |  |
| --- | --- |
| 标识 | 特性 |
| 1 | 对于需要署名登录的网站，它允许创建用户帐号。 |
| 2 | 允许为每个用户存储cookies 和Active Server Pages (ASP) 的session信息 |
| 3 | 支持随机的或顺序的数据集，以用在特定的名字-值对 |
| 4 | 支持带宽调节和随机延迟（“思考的时间”）以更真实地模拟显示情形。 |
| 5 | 支持Secure Sockets Layer (SSL)协议 |
| 6 | 允许URL分组和对每组的点击率的说明 |
| 7 | 提供一个对象模型，可以通过Microsoft Visual Basic® Scripting Edition (VBScript)处理或者通过定制编程来达到开启，结束和配置测试脚本的效果。 |

**2．WAS的缺陷**

除了优势外，WAS的确有一些缺陷存在。当前知道的bug和有关事项都列在WAS的网站上了。以下是当前WAS不支持的特性：

|  |  |
| --- | --- |
| 标识 | 不支持的特性 |
| 1 | 以前面所发请求返回的结果为基础，修改URL参数的能力。 |
| 2 | 运行或模仿客户端逻辑的能力 |
| 3 | 为所分配的测试指定一个确定数量的测试周期的能力。 |
| 4 | 对拥有不同IP地址或域名的多个服务器的同时测试能力 |

注意：

|  |  |
| --- | --- |
| 标识 | 注意 |
| 1 | 你可以使用多个主客户端来同时测试多个服务器。然而，如果你想把所有测试结果联系起来成为一个整体，则需要整理从各个WAS数据库得到的数据 |
| 2 | 支持页面在不同IP地址或域名间的重定向的能力 |
| 3 | 从Web浏览器直接记录SSL页面的能力 |
| 4 | WSA已经支持SSL页面的测试，但是没有记录它们。你需要在脚本录制完后，手工地为每个设计好的URL打开SSL支持 |
| 5 | 虽然对这些限制有一些相应的解决办法，但是如果你的应用依赖一个或多个这样的功能的话，你也许不能完全享受WAS带来的好处。 |

**3．WAS的安装**

|  |  |
| --- | --- |
| 可能出现的问题 | 解决办法 |
| 找不到MSVCP50.dll | 将附件的另一个软件msvcp50.DLL放在C:\Windows\System32下 |
| 没有权限 | 安装好的web application stress tool的可执行文件名hclient.exe右键以管理员身份运行 |

## 1.3开始使用WAS

要对网站进行负载测试首先必须创建 WAS 脚本模拟用户活动。我们可以用下面四种方法之一创建脚本：

|  |  |
| --- | --- |
| 标识 | 方法 |
| 1 | 通过记录浏览器的活动 |
| 2 | 通过导入IIS日志 |
| 3 | 通过把WAS指向Web网站的内容 |
| 4 | 手工制作 |

以下将通过记录浏览器的活动来完成本次实验内容。

1. **录制测试脚本**

虽然你可以手动地创建测试脚本，WAS可以通过记录浏览器活动，导入服务器日志文件或评估Web文件夹的内容来帮助你创建测试脚本。在本文，我们将主要通过记录浏览器活动的方式来创建测试脚本。采用这个方法而不用其它的方法有几个原因，包括：

记录浏览器活动的方式以精确的方式捕捉所有用户的交互活动。任何从浏览器发往服务器的URL指向，应用程序参数和HTTP头部信息都会被自动地记录在新的测试脚本里。

导入服务器日志文件的方法在站点已经进入投入使用阶段，有了真实的用户流量的情况下使用最好。但是，一个新的站点未必有这么多的真实用户使用数据，进一步说，可能还需要合并大量的日志文件来达到较好地体现用户活动的目的，这将需要创建大量的测试脚本，将需要客户端更多的系统资源。

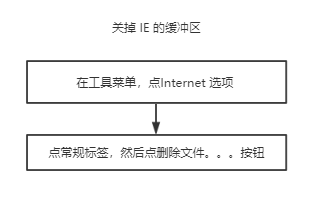
选取Web内容文件夹的方法最好用在测试多数是静态HTML文件的站点。这种方法允许在已有服务器的Web页面的基础上快速创建测试脚本。然而，这种方法并不捕捉任何由大多数应用程序文件产生的参数，像Common Gateway Interface (CGI)程序或Active Server Pages (ASP).

你只需要在主客户机器创建和存储你的测试脚本，当测试由主客户端初始化时，测试脚本会自动地分发到其他的测试客户端。

## 2.1.浏览器准备

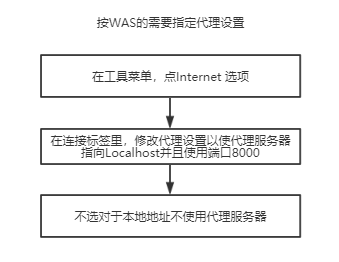
在开始录制一个脚本前，你需要准备好你的浏览器，清除你的浏览器的缓冲cache。否则，WAS也许不能记录所需的浏览器活动，因为浏览器可能从缓冲区而不是从所请求的服务器取得请求页面。

关掉 IE 的缓冲区：



如果使用IE5.0或以上版本则不需要修改代理设置，因为5.0以上版本的IE允许WAS改变这些设置。然而，对于IE4.0或早期版本，WAS使用一个内置的代理服务器来记录浏览器活动。

按WAS的需要指定代理设置：



如果你的浏览器默认不是IE，最好改为IE，同时首页就设置为要测试的系统，否则一旦打开浏览器就去访问一些莫名其妙的网站，访问记录就会多出非常多的杂项。

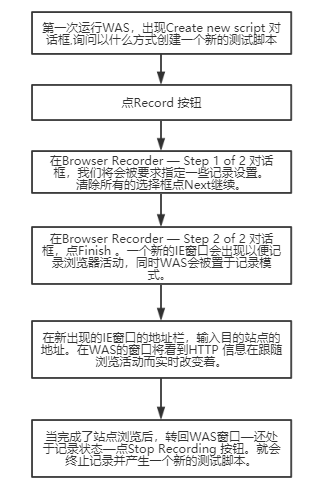
如：将IE浏览器设置为默认浏览器，同时设置百度的首页作为浏览器主页，网址为<http://119.75.218.70/>。（百度ip，域名访问不到）之后清理浏览器的临时文件以及cookie等。





## 2.2记录脚本

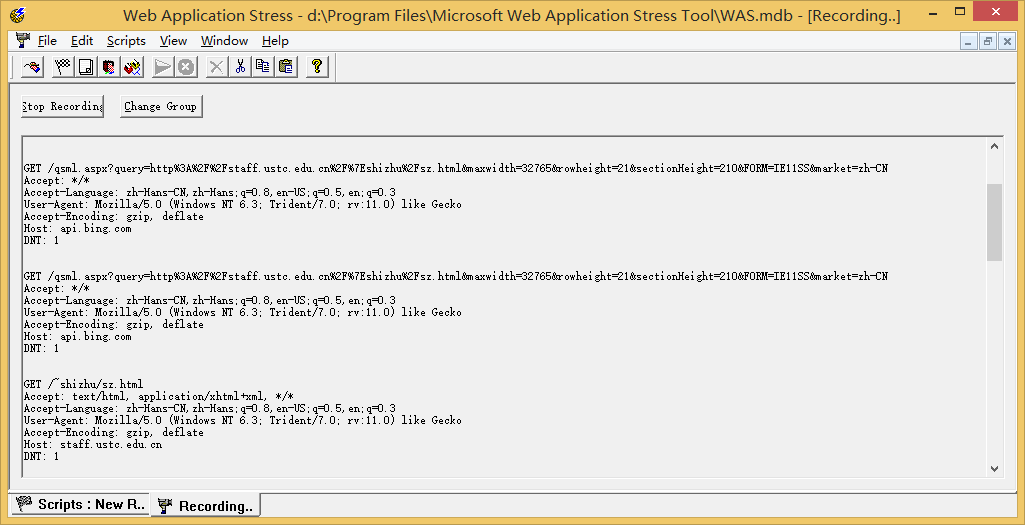
在你的浏览器和客户端已经准备好记录后，做下面的操作：



Record delay between request： 记录了请求之间的延迟。 由于用户实际上在浏览网站时，请求之间存在几秒甚至几分钟的延迟，这种录制方法在执行时会模仿用户之间的延迟发送请求，所以会是一个更加实际的测试。如果我们的目的是要发现 Web 应用程序的承受极限，就不要选择该项；如果只是想模拟一个特定数量的用户场景，那么选择该项进行测试捕捉请求延迟。

Record browser cookies & Record the host header： 只记录用户的会话， 不记录延迟时间。一般情况下，我们不需要选择这两项，可以让 WAS 创建 cookies 和 host header，就好像用户登陆你的网站一样。然而，如果你有网站的回归信息时（比如一个用户的主要特征信息或者与一个永久性 cookies 相连的其他信息），在模拟一个新的用户登陆网站和进行必要的用户配置测试前，必须保证清除 cookies， 如果 Web 应用程序需要用户接受 cookies，那么需要选中该选项。

目前这个版本的 WAS 软件对基于浏览器 IE 录制脚本的方式还不支持 HTTP/SSL 请求。一般情况下，只选择后二种会增加压力的强度。



对于需要安全连接的站点，WAS支持SSL页面。然而不允许SSL的记录。要解决这些限制，你可以在服务器端关掉SSL，记录脚本，然后再重新激活服务器上的SSL。

制作 WAS 脚本是相当简单的，不过要制作出模拟真实用户活动的脚本有点儿复杂。如果你已经有一个运行的 Web 网站，可以使用 Web 服务器的日志来确定 Web 网站上的用户点击分布。如果你的应用还没有开始运行，那么只好根据经验作一些猜测了。

## 2.2设置脚本参数

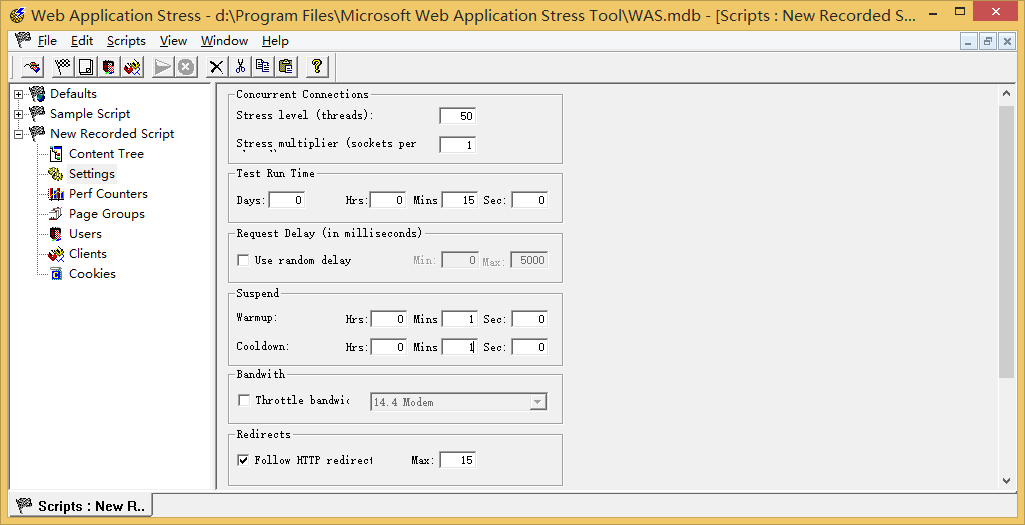
测试脚本录制完成后，下一步要做的就是配置运行脚本的负载选项，可以调整测试配置以便观察不同条件下的应用性能。

### 2.2.1 Content Tree

由于我们的 WAS 和 Web Server 是分开的，所以这里不需要设置

### 2.2.2 Setting（设置）

点击“ Setting”就开始负载选项设置。



1．ConCurrent Connections：

Stress Level（ threads）的数值决定了所有客户机创建的 Windows 的线程的数值。每一个线程创建多个 Socket 连接（具体多少 Socket 连接数取决于 Stress multiplier(sockets per thread)），每个 Socket 连接就是一个并发的请求(request)。下面这个公式表示了它们之间的关系：

总的并发请求数 = Stress Level \* Stress multiplier = 总的 Socket 连接数

Stress Level 和 Stress multiplier 这二个项决定了访问服务器的并发连接的数量。

Microsoft 建议不要选择超过 100 的 Stress Level 值。如果要模拟的并发连接数量超过 100 个， 可以调整 Stress multiplier 或使用多个客户机。 在负载测试期间 WAS将通过 DCOM 与其他客户机协调。

2．Test Run Time：

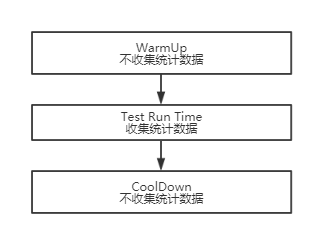
设定持续运行多长时间的测试。我们可以在这里设定让 WAS 持续运行多少天、多少个小时、多少分钟、多少秒

3．Request Delay（ in milliseconds）：

设定请求延迟时间的最大、最小值，当然我们也可以选择“ Use random delay”使用随机的延迟时间。一般情况下，我们常常会浏览一页，发现一个链接后，我们点击它。即便对该网站熟悉的人，

4．Suspend：

WAS 允许设置 warmup （热身）时间，一般可以设置为 1 分钟。在 warmup期间 WAS 开始执行脚本，但不收集统计数据。 warmup 时间给 MTS、数据库以及磁盘缓冲等一个机会来做准备工作。如果在 warmup 时间内收集统计数据，这些操作的开销将影响性能测试结果。 WAS 也允许设置 CoolDown 时间。在 WAS 执行的时间达到设定的 Test Run Time 时，进入 CoolDown Time，这时 WAS 并没有停止执行脚本，同样也不会收集统计数据。下图表示了它们的先后关系。



5．Bandwith：

设置页面提供的另外一个有用的功能是限制带宽（ throttle bandwidth）。带宽限制功能能够为测试模拟出 Modem（ 14.k K， 28.8 K， 56 K）、 ISDN（ 64 K，128 K）以及 T1（ 1.54 M）的速度。使用带宽限制功能可以精确地预测出客户通过拨号网络或其他外部连接访问 Web 服务器所感受的性能。

6．Redirects、 Throughput、 Name resolution：

这几个选项一般情况下采用默认情况即可。选中 Follow HTTP redirects 选项将会支持重定向。选中 Throughput 中的两项， WAS 将会收集活动用户的 cookies，以及收集网站的统计数字。默认情况下都会选中这两项，如果不选择，将会增加压力测试的强度。Name resolution 默认情况下没有选中。选中该选项，会让每一个客户测试机执行查询，只有在使用多个子网时才需要选中该项。（帮助原文： have each individual test client perform a lookup， this is useful when using multiple subnets）

### 2.2.3 Perf Counters（性能计数器）

使用 WAS，从远程 Windows NT 和 Windows 2000 机器获取和分析性能计数器（ Performance Counter）是很方便的。加入计数器要用到下图所示的 Perf Counters 分枝。

一般情况下，这里需要添加的性能计数器有：

|  |  |
| --- | --- |
| 性能计数器 | |
| Web Server： | 处理器： CPU 使用百分比（ % CPU Utilization） |
|  | 内存：内存使用百分比（ % Memory Utilization） |
|  | 线程：每秒的上下文切换次数（ Context Switches Per Second (Total)） |
|  | ASP：每秒请求数量（ Requests Per Second） |
|  | ASP：请求执行时间（ Request Execution Time） |
|  | ASP：请求等待时间（ Request Wait Time） |
|  | ASP：置入队列的请求数量（ Requests Queued） |
| 各个 WAS 测试机： | 处理器： CPU 使用百分比（ % CPU Utilization） |
|  | 内存：内存使用百分比（ % Memory Utilization） |

在测试中选择哪些计数器显然跟测试目的有关。虽然下面这个清单不可能精确地隔离出性能瓶颈所在，但对一般的 Web 服务器性能测试来说却是一个好的开始。

|  |
| --- |
| 处理器： CPU 使用百分比（ % CPU Utilization） |
| 线程：每秒的上下文切换次数（ Context Switches Per Second (Total)） |
| ASP：每秒请求数量（ Requests Per Second） |
| ASP：请求执行时间（ Request Execution Time） |
| ASP：请求等待时间（ Request Wait Time） |
| ASP：置入队列的请求数量（ Requests Queued） |

CPU 使用百分比反映了处理器开销。CPU 使用百分比持续地超过 75%是性能瓶颈在于处理器的一个明显的迹象。每秒上下文切换次数指示了处理器的工作效率。如果处理器陷于每秒数千次的上下文切换，说明它忙于切换线程而不是处理 ASP 脚本。

每秒的 ASP 请求数量、执行时间以及等待时间在各种测试情形下都是非常重要的监测项目。每秒的请求数量告诉我们每秒内服务器成功处理的 ASP 请求数量。执行时间和等待时间之和显示了反应时间，这是服务器用处理好的页面作应答所需要的时间。

我们可以绘出随着测试中并发用户数量的增加每秒请求数量和反应时间的变化图。增加并发用户数量时每秒请求数量也会增加。然而，我们最终会达到这样一个点，此时并发用户数量开始“压倒”服务器。如果继续增加并发用户数量，每秒请求数量开始下降，而反应时间则会增加。要搞清楚硬件和软件的能力，找出这个并发用户数量开始“压倒”服务器的临界点非常重要。

置入队列的 ASP 请求数量也是一个重要的指标。如果在测试中这个数量有波动，某个COM 对象所接收到的请求数量超过了它的处理能力。这可能是因为在应用的中间层使用了一个低效率的组件，或者在 ASP 会话对象中存储了一个单线程的单元组件。

运行 WAS 的客户机 CPU 使用率也有必要监视。如果这些机器上的 CPU 使用率持续地超过 75%，说明客户机没有足够的资源来正确地运行测试，此时应该认为测试结果不可信。在这种情况下，测试客户机的数量必须增加，或者减小测试的 Stress Level。

### 2.2.4 Page Groups

对于一个 Web 应用而言，同一时刻用户点击分布是不一样的。 WAS 允许设置用户点击流量的分布比例。

这里我们假设在一个 Web 应用程序中，有 250 个人同时在线，其中 60 人正在添加提交数据，占 24.00%；有 150 人正在查询，占 60.00%。按照不同的 Web 应用，我们可以根据实际的情况在定制这个比例关系，来更加符合实际的情况。

### 2.2.5 Users

现在很多 Web 应用程序为了提供个性化的服务，都设计了登陆过程。每个用户都有自己的登陆名和密码。 WAS 也考虑到了这种情况，我们只要在 Users 分支中添加用户名和对应的密码即可。

### 2.2.6 Clients

添加多个 WAS 客户机。在运行期间，各个 WAS 客户机是通过 DCOM 来协调的。各个WAS 客户机只要正确安装了 WAS 软件，启动了 WebTool 服务，它们就可以自己协调操作。只要在 Clients 分支内添加 WAS 客户机即可。

### 2.2.7 Cookies

这里显示的是用户名以及对应的 cookies。这里不需要设置。

1. **运行测试脚本**

所有的设置完成以后，我们就可以运行 WAS 来进行压力测试了。要运行测试脚本很简单，只要选中测试脚本的名称，然后点工具栏上的“运行”按钮，即可。建议：第一次运行测试脚本时， Test Run Time 不要太长， Stress Level 以及 Stress multiplier 不要太大。第一次运行的目的只是为了检验测试脚本正确的运行。

1. **运行结果**

## 4.1 摘要

页面摘要部分提供了页面的名字，接收到第一个字节的平均时间（ TTFB），接收到最后

一个字节的平均时间（ TTLB），以及测试脚本中各个页面的命中次数。 TTFB 和 TTLB 这两个值对于计算客户端所看到的服务器性能具有重要意义。 TTFB 反映了从发出页面请求到接收到应答数据第一个字节的时间总和（以毫秒计）， TTLB 包含了 TTFB，它是客户机接收到页面最后一个字节所需要的累计时间。只要选中页面的名字，即可显示页面概要。

## 4.2 Result Codes

如果这是一个新创建的测试脚本，你应该检查一下报表的 Result Codes 部分。这部分内容包含了请求结果代码、说明以及服务器返回的结果代码的数量。如果这里出现了 404代码（页面没有找到），说明在脚本中有错误的页面请求。具体的错误代码表示的意义，可以参考 IIS 的说明文档。

## 4.3 Pref Counters

这一块报表中应该包含了所有性能计数器的信息。这些数据显示了运行时各个项目的测量值，同时还提供了最大值、最小值、平均值等。报表实际提供的信息远远超过了我们这里能够介绍的内容。

## 4.4 Scripts Settings

这里显示的是运行本次测试时的设置，也就是前面讲到的 Setting 部分的内容。

## 4.5 Test Clients

这里显示的是各个 WAS 客户机的情况。先总体说明在测试中使用了那些 WAS 客户机，在使用的 WAS 客户机中显示

|  |
| --- |
| 执行了多少线程 |
| 模拟了多少用户 |
| 点击的次数 |
| 连接失败的次数 |

## 4.6 Page Summary

显示了在测试中各个请求内容的 TTFB 和 TTLB，以及点击的次数等信息。

## 4.7 Page Groups

显示不同的用户组在测试中的执行情况。这里提供的信息包括

|  |
| --- |
| 用户组的分布情况，以及在所有用户组中所占的比例 |
| 点击的次数，以及在所有点击次数中所占的比例 |
| Result Codes 情况 |
| Socket 连接的信息 |

## 4.8 Page Data

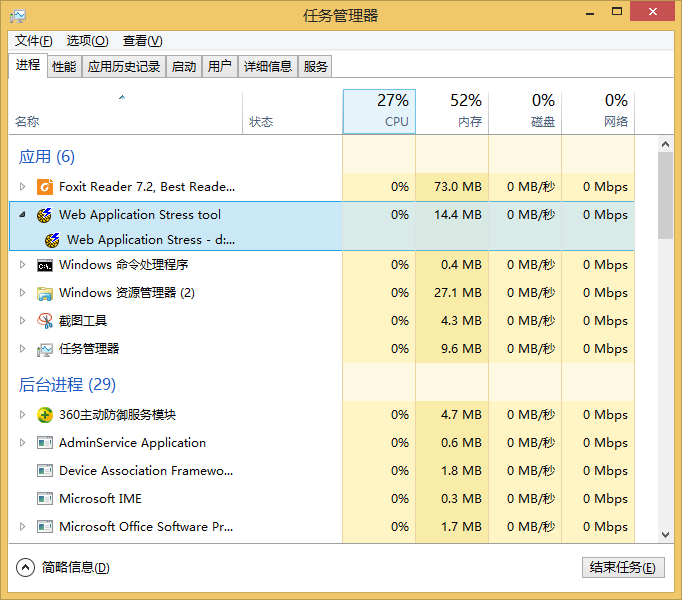
显示了各个请求内容的更加详细的信息。技术需求中的运行效率信息可以在这里验证。

1. **其他检测**

## 5.1 DOS检测

这时可以用NETSTAT –a 来查看是否开启了很多连接，其中state栏established的表示已连接上。

## 5.2 CPU检测



## 5.3 CPU使用率



1. **结论**

基本熟悉了WAS性能测试的整个流程，完成熟悉工具的使用的目的。需更进一步的思考与探索。