https://www.cnblogs.com/zhangchaoyang/articles/2530731.html

[JMeter使用总结](https://www.cnblogs.com/zhangchaoyang/articles/2530731.html)

**使用JMeter做压力测试**

**1．下载Jmeter**

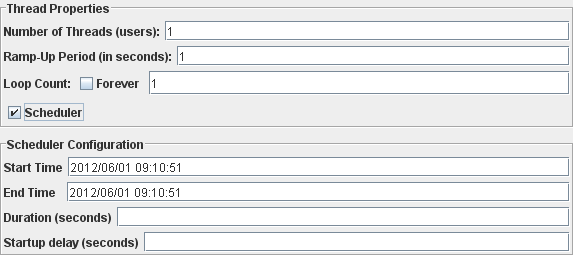
地址：<http://jmeter.apache.org/download_jmeter.cgi>

**2．启动jmeter**

运行bin/jmeter.bat

**3．添加线程组**

在TestPlan节点上右键，Add-->Threads(Users)-->Thread Group。



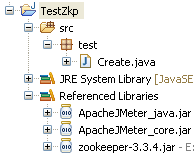
Number of Threads (Users)：要模拟的并发用户量。  
Ramp Up Period (in seconds)：在多长时间内均匀启动所有的线程。比如Number of Threads设为10，Ramp Up Period设为1，则jmeter每隔0.1秒启动1个线程。  
Loop Count：单用户任务重复执行的次数。可以设为Forever，这样jmeter就不会自动停止，需要强制终止。  
还可以设置Scheduler Configuration。这里有两组设置：指定StartTime和End Time让jmeter在特定的时间区段内执行工作；Startup Delay表示从当前时刻开始延迟多长时间开始运行，Duration设定运行时长。

**4．用jmetr向服务器发送Java Request**

当然传统的方法是让jmeter向服务器发送Http Request或FTP Request，但这里不作展示。

假如我们要模拟多用户并发向Zookeeper Cluster发送查询znode的请求，首先要写1个程序来指定单用户的请求行为。

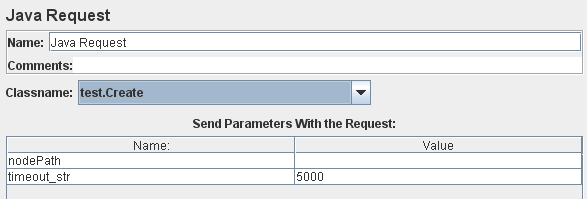
新建一个Java Project，导入jmeter/lib/ext下的两个jar包：ApacheJMeter\_core.jar和ApacheJMeter\_java.jar。



|  |
| --- |
| package test;    import org.apache.jmeter.config.Arguments;  import org.apache.jmeter.protocol.java.sampler.AbstractJavaSamplerClient;  import org.apache.jmeter.protocol.java.sampler.JavaSamplerContext;  import org.apache.jmeter.samplers.SampleResult;  import org.apache.zookeeper.ZooKeeper;    public class Create extends AbstractJavaSamplerClient{      private String nodePath=null;      private String timeout\_str=null;      private SampleResult results;         //设置传入的参数，可以设置多个，已设置的参数会显示到Jmeter的参数列表中      public Arguments getDefaultParameters() {          Arguments params = new Arguments();          params.addArgument("nodePath","");          params.addArgument("timeout\_str", "5000");          return params;      }        //初始化方法，实际运行时每个线程仅执行一次，在测试方法运行前执行      public void setupTest(JavaSamplerContext arg0) {          results = new SampleResult();          nodePath = arg0.getParameter("nodePath");   //获取在Jmeter中设置的参数值          if (nodePath != null && nodePath.length() > 0) {              results.setSamplerData(nodePath);          }          timeout\_str=arg0.getParameter("timeout\_str");          if (timeout\_str != null && timeout\_str.length() > 0) {              results.setSamplerData(timeout\_str);          }      }         //测试执行的循环体，根据线程数和循环次数的不同可执行多次      @Override      public SampleResult runTest(JavaSamplerContext arg0) {          int timeout=Integer.parseInt(timeout\_str);          boolean success=true;          results.sampleStart();      //事务的起点          try {              ZooKeeper zkp=new ZooKeeper("localhost:2181",timeout,null);              zkp.getData(nodePath, false, null);              zkp.close();          } catch (Exception e) {              success=false;          }finally{              results.sampleEnd();    //事务的终点              results.setSuccessful(success);     //设置本次事务成功或失败          }          return results;      }        //结束方法，实际运行时每个线程仅执行一次，在测试方法运行结束后执行      public void teardownTest(JavaSamplerContext arg0) {      }  } |

将project导出为jar包，放到jmeter/lib/ext下，重启jmeter。

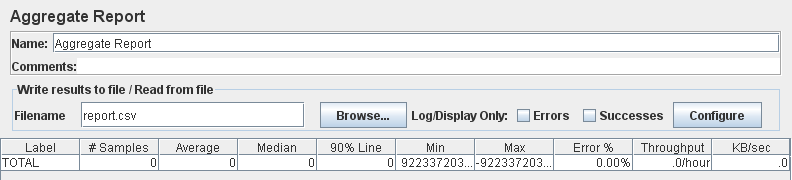
在Thread Group上是右击，Add-->Sampler-->Java Request。



设置nodePath和timeout的值。

**5．添加Listener**

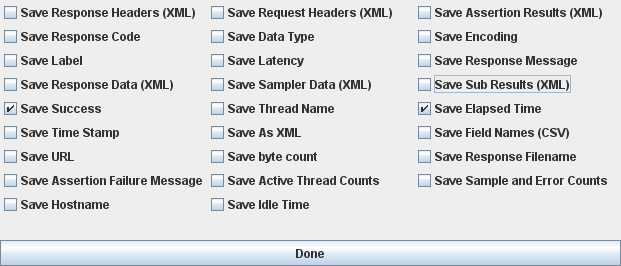
在TestPlan上右击，Add-->Listener-->Aggregate Report。Jmeter生成的报告有多种，这里以Aggregate Report为例。



可以指定将report保存到某个文件，如上图中所示保存到report.csv，在非GUI模式下运行Jmeter这是非常必要的。

可以在bin/jmeter.properties中设置report文件的默认格式，你可能要经常要更改这两项配置：jmeter.save.saveservice.output\_format=csv #默认有文件格式为jtl jmeter.save.saveservice.default\_delimiter=\t #默认使用","作为域之间的分隔符

点击Configure可以自己定义report中你所关心的数据项。



比如我只关心每个请求是否成功，以及每个请求所用的时间，所以只勾选了两项。

**6．为Jmeter安装PerfMon插件**

PerfMon用来监控Server的CPU、I/O、Memory等情况。

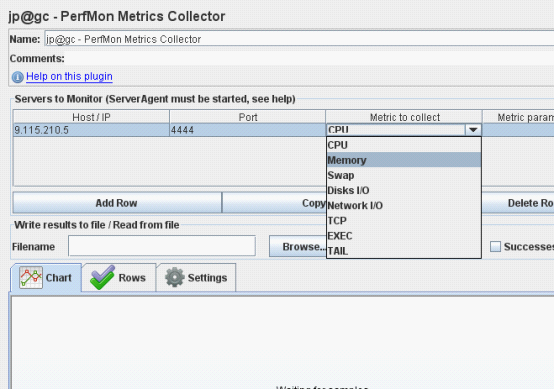
1. 插件下载地址：<http://code.google.com/p/jmeter-plugins/wiki/PerfMon>

2. 把JMeterPlugins.jar放到jmeter客户端的jmeter/lib/ext下。

3. 启动jmeter，添加Listener时你就看到PerfMon Metrics Collectors了。

4. 另外还需要把下载下来的PerfMon解压后放到所有的被测试的服务器上，并运JMeterPlugins/serverAgent/startAgent.sh，默认工作在4444端口。

5. 使用PerfMon截图：



再次提醒一下，在非GUI模式下运行Jmeter时指定把result保存到一个文件是非常必要的。

**7．非GUI模式下运行Jmeter**

具体方法是：先在GUI模式下创建TestPlan，保存为jmx文件。命令行启动jmeter：./ApacheJMeter -n -t testplan.jmx (选项-n表示non-GUI，-t指定TestPlan文件)。运行结束后Aggregate Report和PerfMon Metrics Collector就会保存在你指定的文件中。把保存PerfMon Metrics Collector的文件拖到Jmerter GUI中就可以看到CUP等使用状况拆线图了。

**8．远程启动jmeter**

应用进场景：用一台机器（称为JMeter客户端）上的jmeter同时启动另外几台机器（称为JMeter远程服务器）上的jmeter。

1. 保证jmeter客户端和jmeter远程服务器采用相同版本的jmeter和JVM。

2. jmeter客户端和jmeter远程服务器最好在同一个网段内。

3. 在jmeter远程服务器上运行JMETER\_HOME/bin/jmeter-server （UNIX）或者JMETER\_HOME/bin/jmeter-server.bat（Windows）脚本 。

4. 在jmeter客户端上修改/bin/jmeter.properties文件，找到属性"remote\_hosts"，使用JMeter远程服务器的IP地址作为其属性值。可以添加多个服务器的IP地址，以逗号作为分隔。   
例如：  
#remote\_hosts=127.0.0.1  
remote\_hosts=9.115.210.2:1099,9.115.210.3:1099,9.115.210.4:1099  
# RMI port to be used by the server (must start rmiregistry with same port)  
server\_port=1099

5. 在jmeter客户端上启动jmeter:  
./jmeter -n -t plan.jmx -r #选项-r表示远程启动(remote)  
jmeter客户端会自动向jmeter远程服务器上分发测试计划。

原文来自:博客园（华夏35度）http://www.cnblogs.com/zhangchaoyang 作者:Orisun

分类: [Java](https://www.cnblogs.com/zhangchaoyang/category/256481.html)