**WebSocket性能测试方案**

# 测试目的

检查Z项目等系统在多用户在线并发使用时，对系统的压力情况分析，根据测试数据计算出系统的性能参数，进行系统优化与系统扩容，保证系统的稳定性与连续性。

# 测试客户机配置

原则上，服务器与客户端的机器应该是1：1配比关系，即两台服务器作为压测服务器时，对应的测试客户端也应该为两台等配或者半配的机器，这里暂定为N台客户端，配置如下：

*\*待补充完整*

|  |  |  |  |
| --- | --- | --- | --- |
| 客户端机器名称 | 硬件配置 | 操作系统 | 软件配置 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 服务器配置

## 3.1测试环境网络拓扑图

*\*待补充完整*

|  |  |  |  |
| --- | --- | --- | --- |
| 网络类型 | 带宽 | 设备 | 数量 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

被测试系统性能环境是模拟环境，网络部署如下，待补充

## 3.2测试环境数据库服务器参数

数据库服务器2N台，分别为主从或者灾备，配置如下：

*\*待补充完整*

|  |  |  |  |
| --- | --- | --- | --- |
| 服务器名称 | 硬件配置 | 操作系统 | 软件配置 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 3.3测试环境后台服务器参数

应用服务器N台，配置如下：

*\*待补充完整*

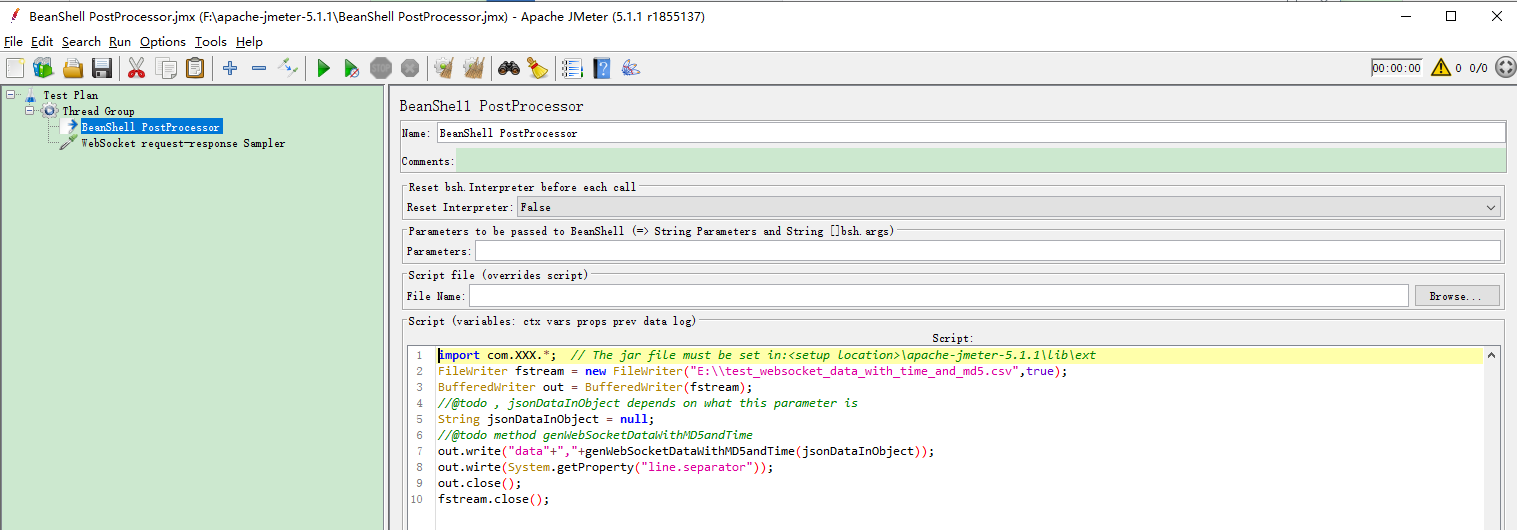
|  |  |  |  |
| --- | --- | --- | --- |
| 服务器名称 | 硬件配置 | 操作系统 | 软件配置 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 测试工具

## Jmeter

### 4.1.1压测脚本编写

* 确认WebSocket 数据包的加密逻辑
* 根据加密逻辑，编写Jmeter压测脚本，如下图



* 待补充

### 4.1.2压测参数配置

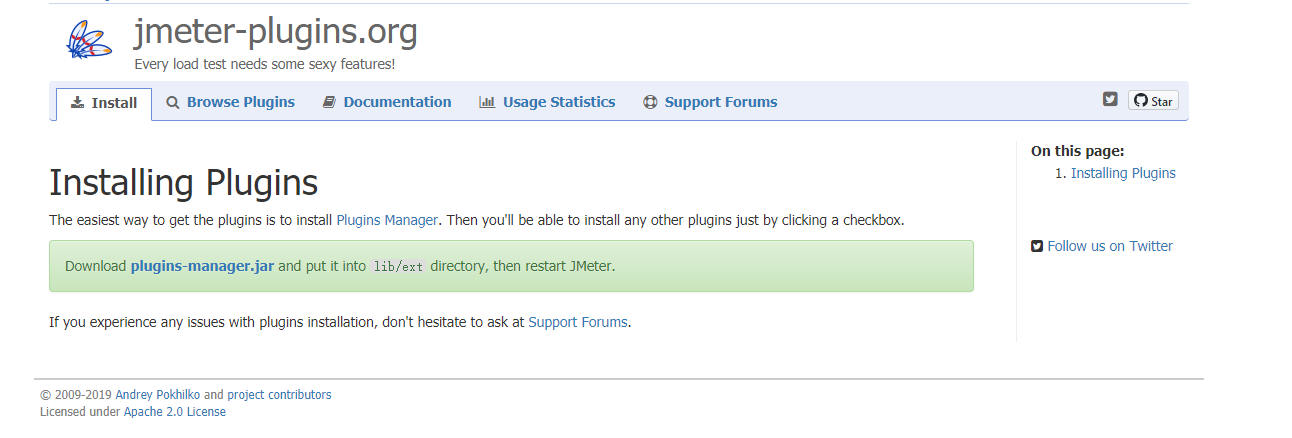
#### 4.1.2.1 使用WebSocket插件

*\*待补充完成，如使用BeanShell Sampler等*

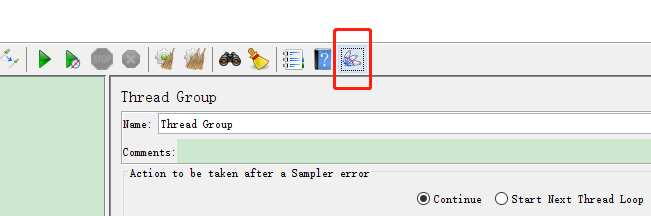
* 启动Jmeter

在win下，点击bin目录下面的Jmeter.bat，在linux下，点击bin目录下面的Jmeter.sh文件

* 安装插件<https://jmeter-plugins.org/install/Install/>



下载plugins-manager.jar 后将其放到Jmeter安装目录中的 lib/ext，并重启Jmeter,然后点击plugins manager图标，



再选择 Available Plugins 标签页，并且点击应用与和重启后，会在Sampler中加入WebSocket供选择

* 添加websocket Sampler

添加线程组，选择Websocket Open Connection模板，如下图

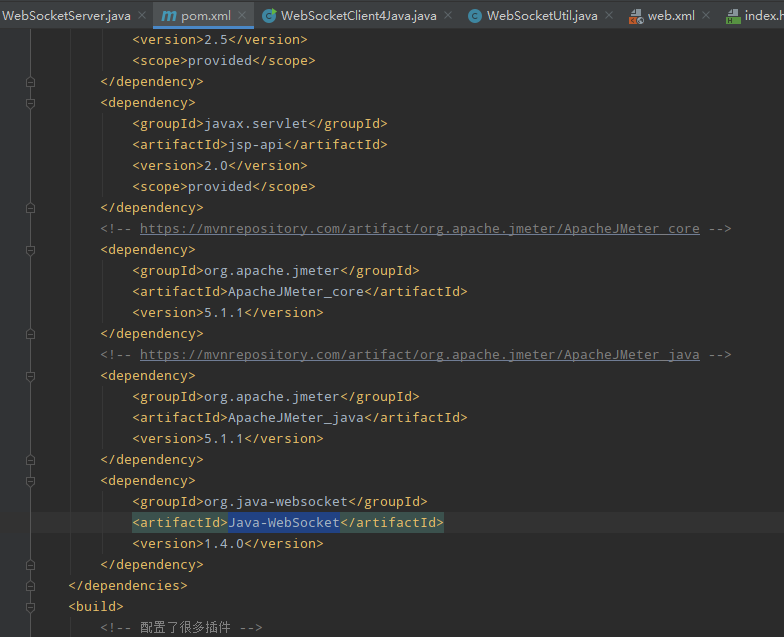


其中，WebSocket Open Connection模板适用于只连接不发送具体data数据的，而WebSocket request-response Sampler是连接后发送data数据，或者只是发送数据等，按具备业务测试场景调用相应的WebSocket模板即可

* 配置过程

#### 4.1.2.2 使用Java Request编程方式

* 创建pom.xml，包括ApacheJMeter\_core、ApacheJMeter\_java和Java-WebSocket

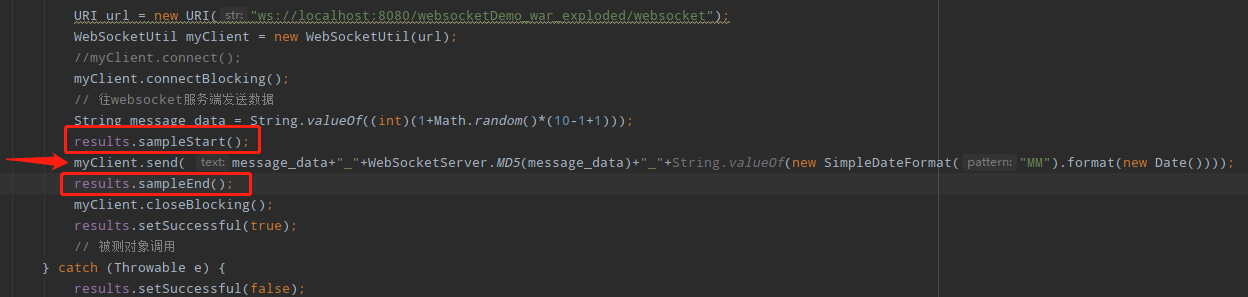


* 编写WebSocket服务端，WebSocketServer.java

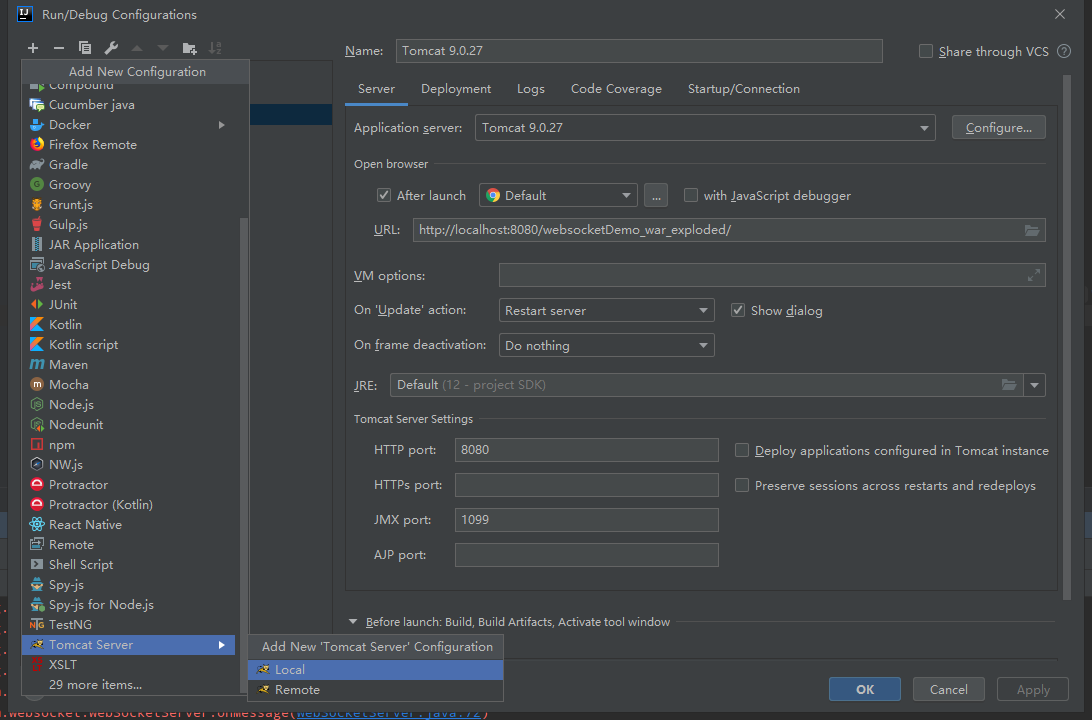
代码请查阅附录中的websocket-master，服务器加入了初步的判断逻辑，如部分message内容的md5校验与时间的校验，通过校验才会sendMessage，具体代码详细请查阅onMessage()方法

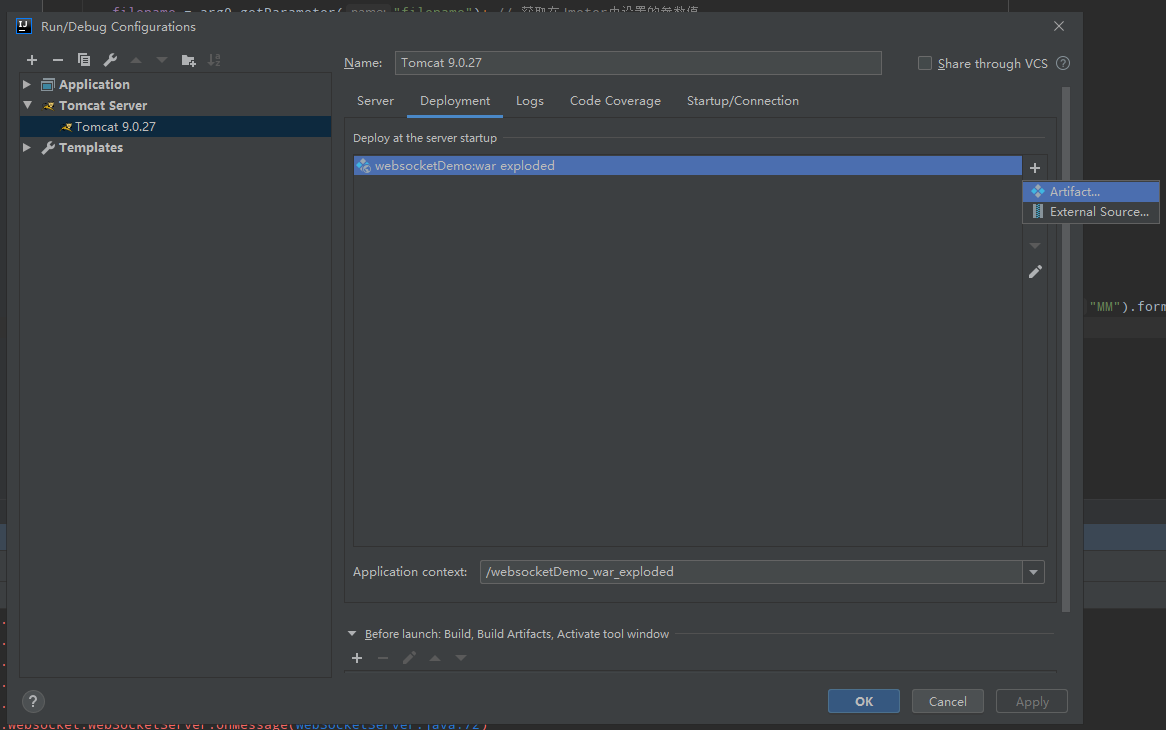
* 编写WebSocket客户端，WebSocketClient4Java.java

代码请查阅附录中的websocket-master，这里的客户端代码兼容了Jmeter的Java Request代码规范，其中压测脚本主要逻辑必须编写在runTest()方法内，另外，在send之前必须做前置的connect，同时发送完成后也必须进行close。最后，针对Jmeter的事务统计的精确度，sampleStart()与sampleEnd()尽量放在压测的最核心逻辑块前后，不过也有特殊情况，像RSA加密时，在具备压测场景也要当作是事务的一部分，那sampleStart()与sampleEnd()的计算范围要扩大，如下图

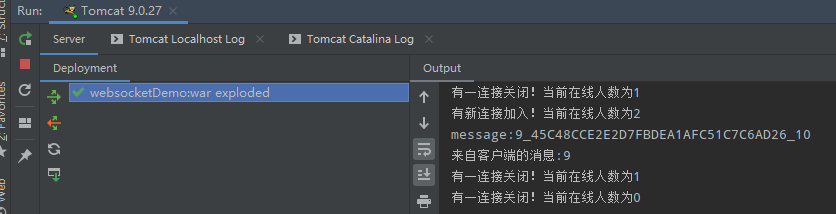


* 配置tomcat,点击Run，再点击Edit Configurations，点击左上角加号，选择tomcat（前提本地安装成功tomcat9），再选择local，点击确定后，出现Tomcat Server,点击右上角的Configure按钮，选择tomcat9的安装目录，成功后，再填写URL为<http://localhost:8080/websocketDemo_war_exploded/，JRE>默认选择jdk12,再点击标签页中的Deployment,跳转到Deploy at the server startup内容页面后，点击右边的加号，点击Artifact Exploded（为什么要选择Exploded，因为方便热部署），最后点击确认。这样run按钮菜单会增加 Run ‘Tomcat 9.X.X’

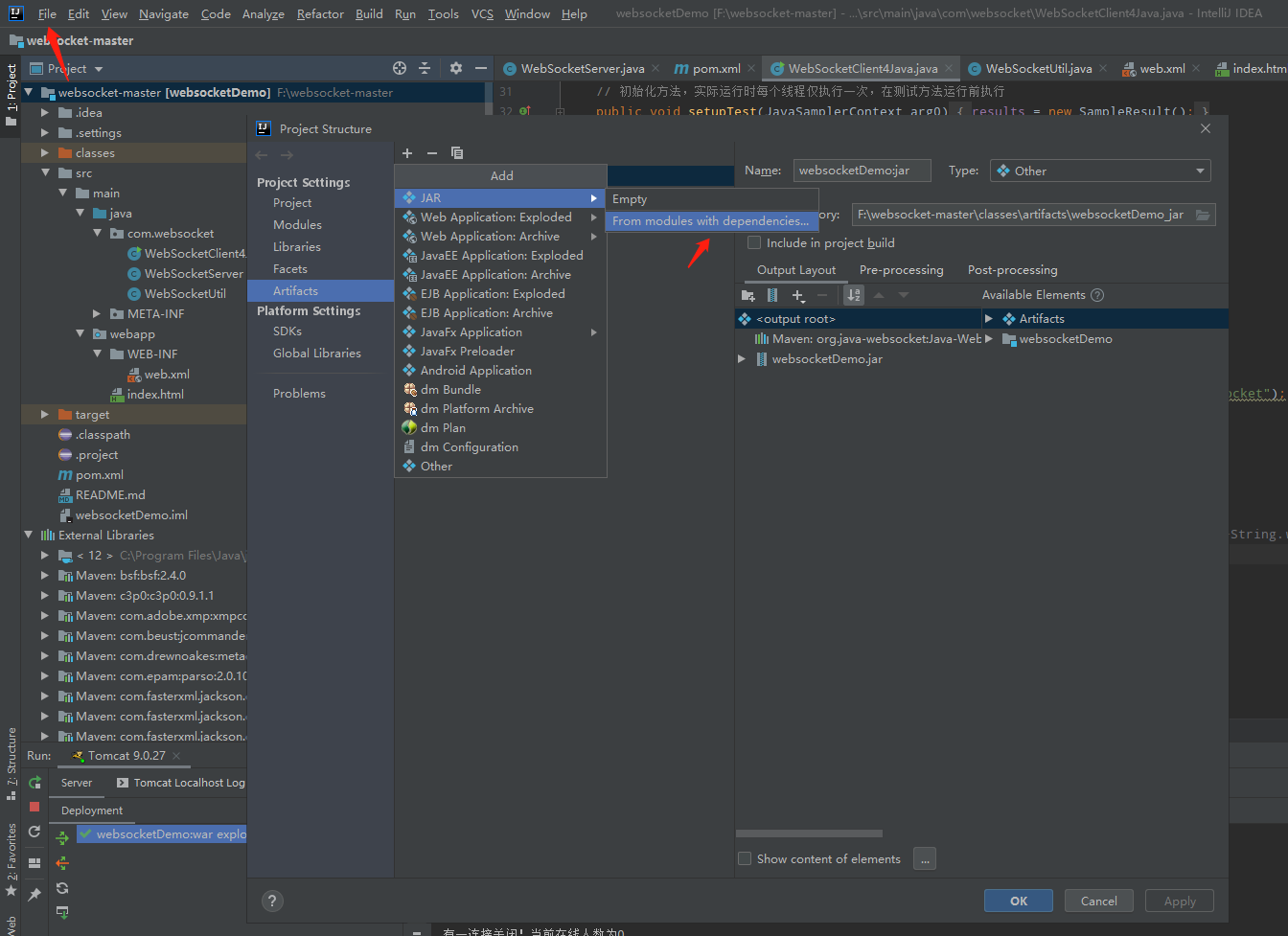




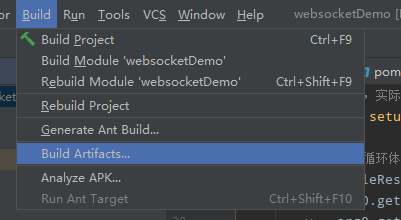
* 点击Run ‘Tomcat 9.X.X’，成功部署的话，左下角可以看到带绿色勾的websocketDemo:war exploded

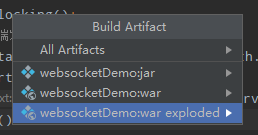


* 客户端代码打包，打开websocket-master项目(为了方便演示，现在客户端与服务器代码都在相同的项目里面)，点击File中的Project Structure中的Artifacts，点击左上角的加号，选择JAR中的From modules with dependencies，在弹出窗口里选择 copy to the output directory and link via manifest点击Ok按钮。然后回到窗口，把<output root>目录下的Maven只保留org.java-websocket这个包，同时系统会自动生成websocketDemo.jar接着点击Ok。如果过程中出现警告提示框，说XXX.mf already exist in VFS，请把F:\websocket-master\src\main\java中的META-INF目录删除，并且重启IDEA就可以解决。另外这里要记录这个地址F:\websocket-master\classes\artifacts\websocketDemo\_jar，因为是生成jar的地址。

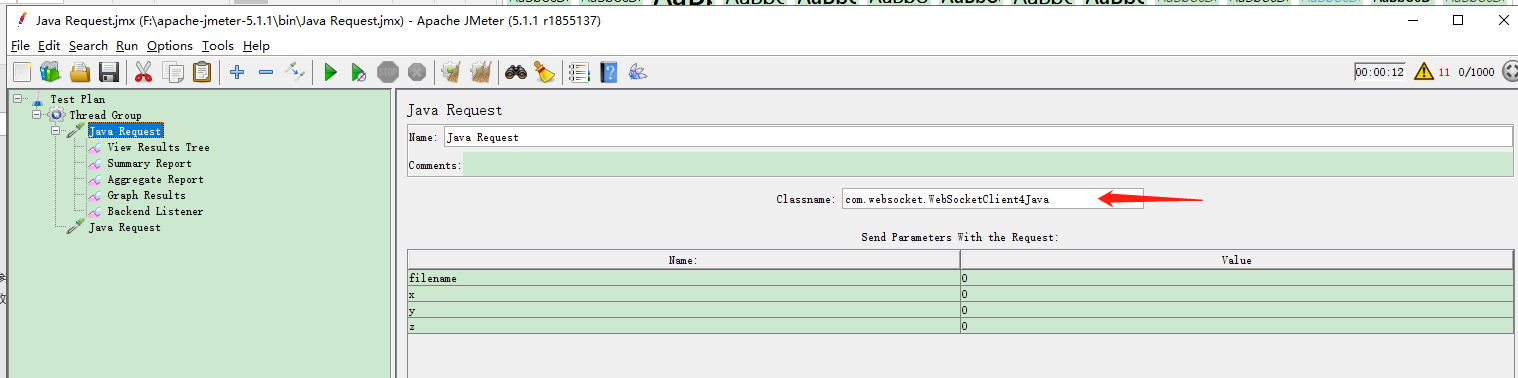


* 点击Build中的Build Artifacts,在弹出的下拉菜单选择 websocketDemo:war exploded，再选择Build，成功后，进入F:\websocket-master\classes\artifacts\websocketDemo\_jar的目录提取Java-WebSocket-1.4.0.jar与websocketDemo.jar





* 将Java-WebSocket-1.4.0.jar与websocketDemo.jar存放到Jmeter安装目录中的lib/ext目录下
* 启动Jmeter与新建Java Request成功后，会看到右边窗口出现客户端的java类名

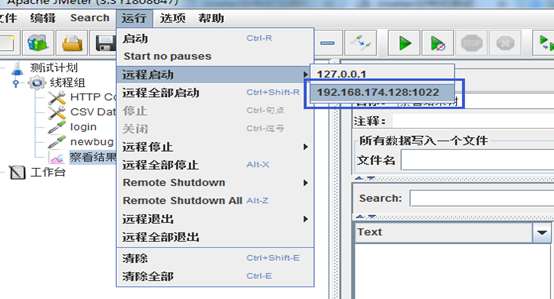


* 详细jmx文件，请到附录下载

### 4.1.3负载机集群配置

* 第一步：在其他负载机上启动jmeter-server.Bat（jmeter-server.Bat windows）(jmeter-server.sh mac ,linux)
* 第二步：主控机修改jmeter.properties文件中的这一句，添加多个ip地址往后面加就可以了，端口号默认1099,如remote\_hosts=127.0.0.1,192.168.1.2:1099，默认jmeter-server的端口号是1099，如果说负载机修改了端口号那么主控机里面的端口也得改。

主控机可以在这里选择负载机：



* Jmeter 在linux运行操作，进入jmeter/bin目录下执行： ./jmeter -n –t /root/bugfree.jmx -l /root/bugfree.jtl

其中参数解释如下：

-n no gui 没有界面的运行

-t 指定一个运行的脚本

-l 指定测试结果的文件

* Jmeter部署在window时，一般是使用Jmeter图形界面进行操作

其中，connections有两个选项，第一个选择已存在的连接配置与第二个新建的连接配置，另外Server Url可以配置ws协议与wss协议（加密通道的协议），最后Data项，可以选择Text与Binary发送，包括响应超时时间配置，而请求参数编写时

## Locust

基于Python语言开发的，暂时不支持WebSocket，但可以自己扩展去支持WebSocket，作为备选压测工具

## LoadRunner12

这个版本LR支持WebSocket协议，但没有破解，只支持最大的VUser为50,也作为备选压测工具

# 测试场景

*\*待补充完整,建议根据系统全功能规范文档，再选择测试项目*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试项编号** | **测试场景** | **测试业务点** | **测试策略** | **测试过程描述** |
|  | 登录  聊天 |  | 基准检测  并发检测  极限检测 | 单用户，20用户，40用户与极限60用户，并发执行共30分钟 |
|  | 战报  征战 |  | 吞吐量检测  场景检测 | 20用户在线综合场景：8用户查看战报，12用户参与征战，持续稳定检测30分钟 |
|  | 竞技场 |  | 稳定性测试 | 单用户在系统加载一定业务压力下，进行竞技场测试，持续24小时 |
|  |  |  | 不适用 | 不适用 |

# 输出参考指标与通过指标

Jmeter的输出报告可以给出如下参数，其中Average、Throughput与Error%是重要的参考指标：

Average：请求的平均响应时间。Request的平均响应时间

Median：中位数，也就是50%用户的响应时间

90%Line：90%用户的响应时间

Min：最小的响应时间

Max：最大的响应时间

Error%：错误率，本次测试中出现错误的请求的数量/请求的总数

Throughput：吞吐量。默认情况下表示每秒完成的请求数，吞吐量=请求数/总时间

Received KB/sec：每秒从服务器端接收到的数据量，即：收到的千字节每秒的吞吐量测试

Sent KB/sec：每秒从客户端发送的请求的数量，即：发送的千字节每秒的吞吐量测试KB/sec：每秒数据包流量，单位是KB

通过指标可以从下面两个指标去判断本次测试是否通过：

* 性能测试场景通过，并满足并发超过500、响应时间低于1秒等要求
* 系统资源消耗
* 服务器CPU利用率<80%
* 物理内存利用率<80%
* 场景通过率>99.99%
* 长时间运行系统，保证系统不崩溃同时功能正常使用
* 测试结束后限定时间内占用资源可以正常释放

# 测试策略

对Z项目业务系统性能测试的主要目的是验证系统是否满足上线业

务运行的性能需求。

根据要求确定性能测试的五个检测项，分别是XX、XX、XX、XX、

XX、充值，根据系统实际业务开展情况，结合系统典型业务并依据频繁的用户操作、大数据量处理等原则，此次性能测试采取XX、XX和日XX三个检测项。

测试内容包括以下三个方面：对游戏业务采取性能指标值中正常情况下的用户数作为基准测试的虚拟用户数；对其游戏业务和战报等查询业务采取随着用户数的递增进行并发测试、极限测试；采取综合场景（充值、征战）的吞吐量测试，持续特定时间。在执行完以上测试后，运行战报查询服务。在执行所有测试期间，尤其是对于极限测试和综合场景吞吐量测试的过程中，检验测试应用系统和硬件的自恢复能力。

概括体现在如下：

* 关键资源不处于阻塞状态

A、服务器CPU利用率<80%

B、物理内存利用率<80%

C、场景通过率>99.99%

* 组合多个场景并发测试
* 测试执行具体情况

采用数量倍增方式，并发数按照50、100等逐步增加，直至在某个具体并发数值增加后TPS达到峰值，并再增加并发造成响应时间增加，事件通过率降低，这里记录此压测参数作为系统性能优化的参考值。

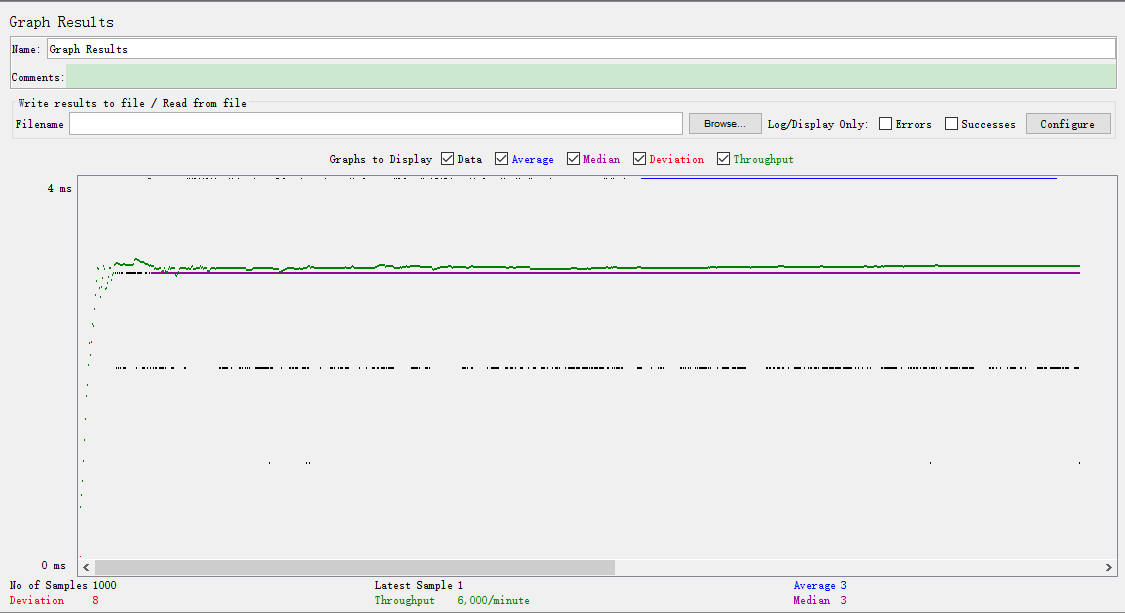
# 测试结果及分析

## 8.1 测试结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **场景设定** | | **总体情况** | | | | **服务器资源** | |
| 检测场景 | 检测业务点 | 并发用户/TPS | 并发成功率(%) | 平均响应时间（秒） | 成功率（%） | CPU利用率（平均%/最高%） | 内存使用情况 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

## 8.2 结果分析

XXX测试结果图



从以上图表分析可得：

测试完毕后，CPU 使用率有所下降，空闲内存略微上涨。此结果说明，服务器性能优良，可以承受来自系统处理的业务场景设计。

## 8.3总结描述

从测试结果看，共有 0 个严重性问题、0 个一般性问题、0 个建议性问题，共计 0 个问题。经优化后，其中 0 个严重性问题、0 个一般性问题、0 个建议性问题已经得到解决。

## 8.4测试问题列表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **问题编号** | **问题描述** | **问题等级** | **检测项编号** | **整改状态** |
|  |  |  |  |  |  |

# 附录

JMeter项目



JMeter的第三方插件库pluign manager

