

目录

- 1. JMeter压力测试
 - 1.1 测试过程
 - 1.2 Linux top命令
- 2. 自定义配置文件JMeter压测
 - 2.1 测试过程
- 3. Redis压测
- 4. Linux环境下，命令行压测
 - 4.1 打成jar包
 - 4.2 上传到Linux服务器上
 - 4.3 编写.jmx文件
- 5. SpringBoot 打war包

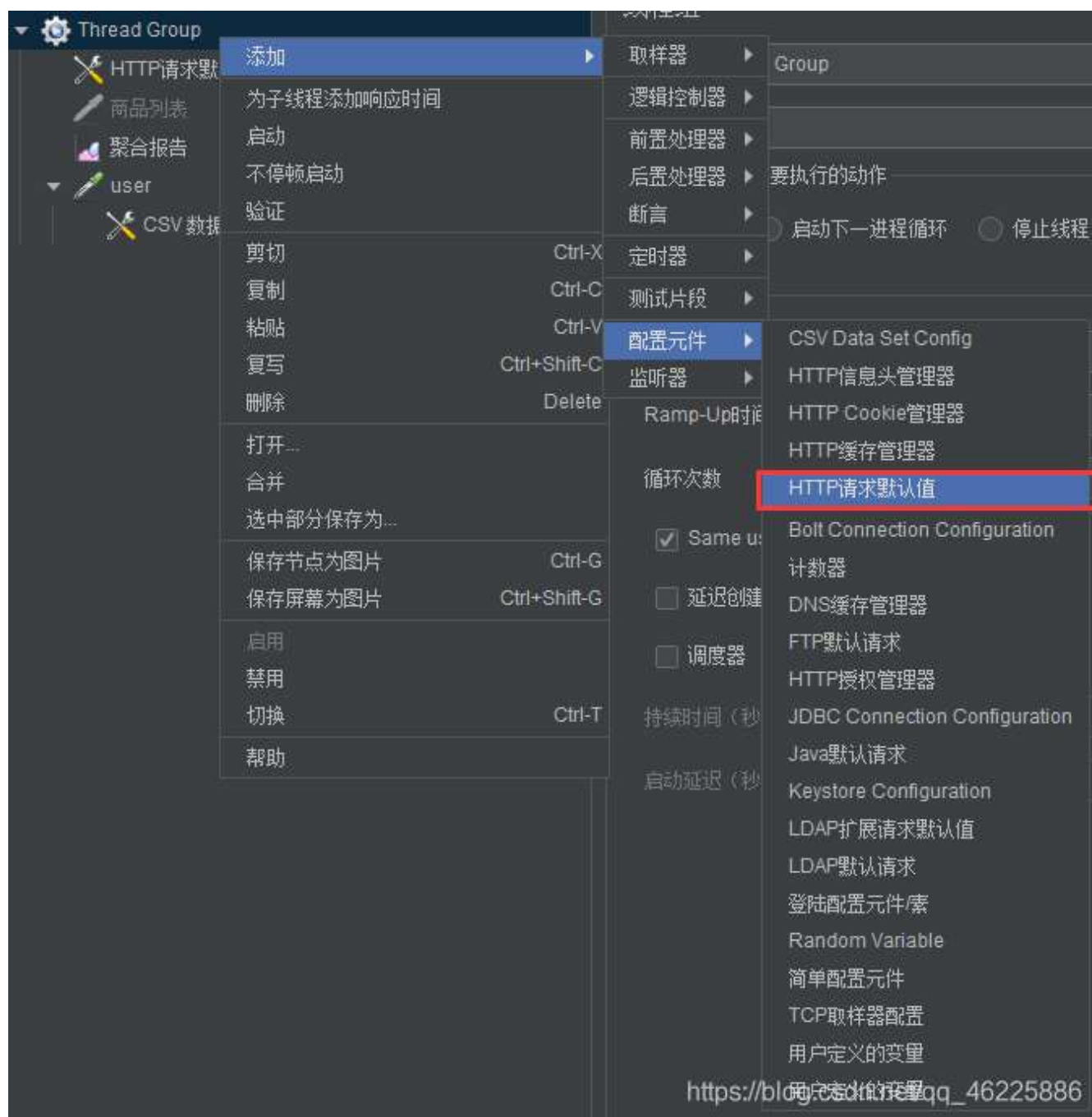
1. JMeter压力测试

1.1 测试过程

- 1. 打开jmeter.bat



- 2. 设置HTTP默认请求



编写协议和端口号

协议:	http	服务器名称或IP:	localhost	端口号:	8080
-----	------	-----------	-----------	------	------

3. 编写测试HTTP请求

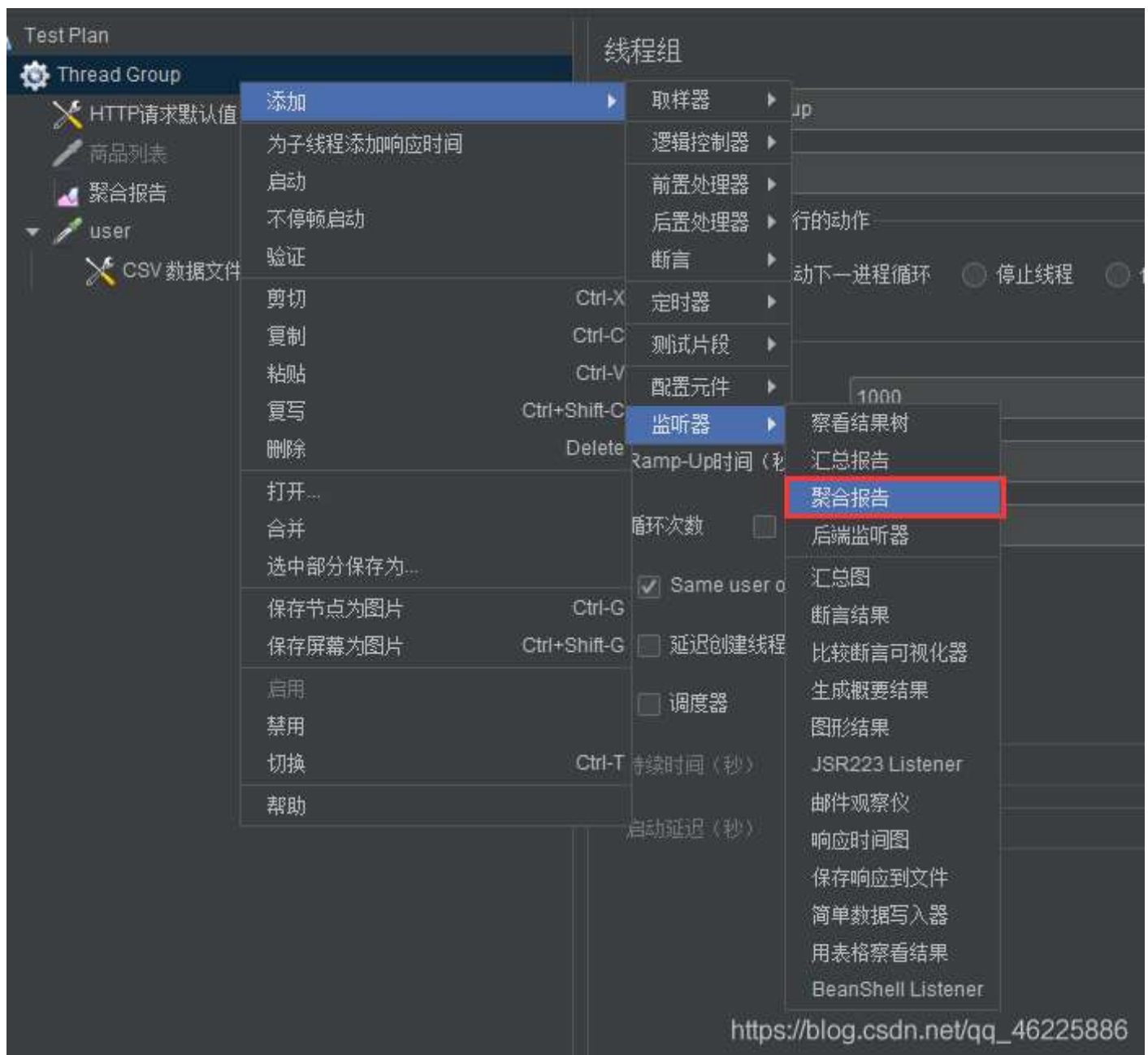


因为我们已经写过 **默认设置**，我们就可以不用编写协议和地址了，如下，只需编写 **请求类型** 和 **地址** 即可

The image shows a screenshot of the 'Web 服务器' (Web Server) configuration dialog box in JMeter. The '协议' (Protocol) is set to 'HTTP'. The '服务器名称或IP' (Server name or IP) is empty. The '端口号' (Port number) is set to '80'. The 'HTTP 请求' (HTTP Request) section shows 'GET' selected in the dropdown, and the '路径' (Path) is '/goods/to_list'. The '内容编码' (Content encoding) is empty. There are checkboxes for '自动重定向' (Automatic redirect), '跟随重定向' (Follow redirect), '使用 KeepAlive' (Use KeepAlive), '对POST使用multipart / form-data' (Use multipart / form-data for POST), and '与浏览器兼容的头' (Compatible with browser headers). The '参数' (Parameters) tab is selected, showing a table for '同请求一起发送参数' (Parameters sent with request). The table has columns for '名称' (Name), '值' (Value), and '编码?' (Encoding?). The '内容编码' (Content encoding) is set to 'UTF-8'. The URL 'https://blog.csdn.net/qq_46225886' is visible in the bottom right corner.

4. 添加聚合报告

我们即可在报告中查看压测信息



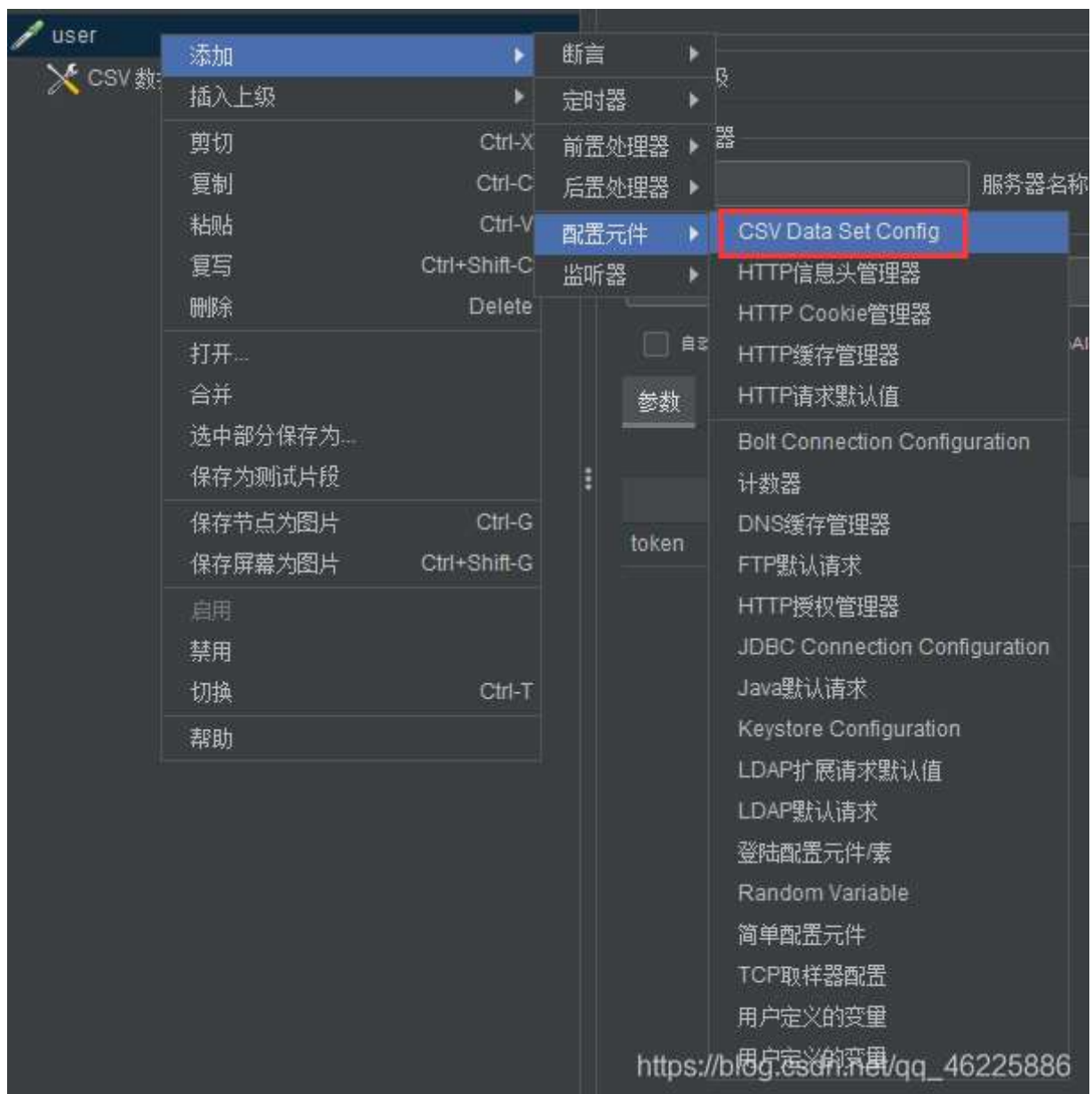
1.2 Linux top命令

- top: 相当于Windows下的任务管理器，可以动态显示当前进程的状况

2. 自定义配置文件JMeter压测

2.1 测试过程

与上方基本一致，不过，要在测试的请求上，[添加CSV数据文件设置](#)



读取我们自己编写的配置文件，并且标注变量名称，如此，即可开始压测。

CSV 数据文件设置

名称: CSV 数据文件设置

注释:

设置 CSV 数据文件

文件名: C:/Users/37407/Desktop/config.txt 浏览...

文件编码:

变量名称(英文逗号分隔): **userId,userToken**

忽略首行(只在设置了变量名称后才生效): False

分隔符(用 \t 代替制表符): ,

是否允许带引号?: False

遇到文件结束符再次循环?: True

遇到文件结束符停止线程?: False

线程共享模式: 所有现场

https://blog.csdn.net/qq_46225886

其中配置文件信息，用英文逗号隔开



3. Redis压测

```
1  #100个并发连接, 100000个请求
2  redis-benchmark -h 127.0.0.1 -p 6379 -c 100 -n 100000
3
4  #存取大小为100字节的数据包
5  redis-benchmark -h 127.0.0.1 -p 6379 -q -d 100
6
7  #测试set和lpush命令的QPS, 其中-q为简化输出
8  redis-benchmark -t set,lpush -q -n 1000000
9
10 #测试单条命令的QPS
11 redis-benchmark -n 100000 -q script load "redis.call('set','foo','bar')"
```

4. Linux环境下, 命令行压测

1. 在Windows目录下写好 `jmx` 文件
2. 命令行: `sh jmeter.sh -n -t xxx.jmx -l result.jtl`
3. 再将result.jtl 导入到jmeter中

4.1 打成jar包

```
1  maven clean package
```

打开jar包, 我们进入META-INF目录下, 打开MANIFEST.MF文件, 我们可以发现如下语句


```
1 Manifest-Version: 1.0
2 Spring-Boot-Classpath-Index: BOOT-INF/classpath.idx
3 Implementation-Title: miaosha
4 Implementation-Version: 0.0.1-SNAPSHOT
5 Start-Class: com.imooc.miaosha.MiaoshaApplication
6 Spring-Boot-Classes: BOOT-INF/classes/
7 Spring-Boot-Lib: BOOT-INF/lib/
8 Build-Jdk-Spec: 1.8
9 Spring-Boot-Version: 2.3.1.RELEASE
10 Created-By: Maven Jar Plugin 3.2.0
11 Implementation-Vendor: Pivotal Software, Inc.
12 Main-Class: org.springframework.boot.loader.JarLauncher
13
```

其中Main-Class为SpringBoot框架的启动类，在这个类中可以跟进看源码
Start-Class为我们自己编写的启动类

4.2 上传到Linux服务器上

```
1 #执行如下命令，之后即可根据如下地址访问
2 #http://182.92.xxx.xxx:8080/login
3 java -jar miaosha.jar
```

4.3 编写.jmx文件

在Windows上用JMeter 编写.jmx脚本，上传到服务器上，执行如下命令行

```
1 jmeter.sh -n -t good_list.jmx -l result.jtl
```

之后，下载result.jtl到Windows本地，进行报告分析

5. SpringBoot 打war包

1. 在pom.xml文件中，添加打包为war包的标签

```
1 <packaging>war</packaging>
```

2. 添加tomcat provided编译时的依赖

```
1 <dependency>
2 <groupId>org.springframework.boot</groupId>
```

```
3         <artifactId>spring-boot-starter-tomcat</artifactId>
4         <scope>provided</scope>
5     </dependency>
```

3. 在主类中，实现SpringBootServletInitializer，重写configure() 方法

```
1  @SpringBootApplication
2  public class MiaoshaApplication extends SpringBootServletInitializer {
3
4      public static void main(String[] args) {
5          SpringApplication.run(MiaoshaApplication.class, args);
6      }
7
8      @Override
9      protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {
10         return builder.sources(MiaoshaApplication.class);
11     }
12 }
```

4. 将ROOT目录删除，并且把我们的war包修改为ROOT.war，放在webapps目录下，即可访问

Name	Date modified	Type
docs	2/6/2018 9:42 PM	File folder
examples	2/6/2018 9:42 PM	File folder
host-manager	2/6/2018 9:42 PM	File folder
manager	2/6/2018 9:42 PM	File folder
ROOT	10/26/2018 2:45 PM	File folder
ROOT.war	10/26/2018 2:33 PM	WAR File