**性能测试**

**概念**

性能测试是通过自动化的测试工具模拟多种正常、峰值以及异常负载条件来对系统的各项性能指标进行测试。[负载测试](http://baike.baidu.com/view/651437.htm)和[压力测试](http://baike.baidu.com/view/565536.htm)都属于性能测试，两者可以结合进行。

**目的**

目的有两个：一个是找出目前系统能承受的最大压力，第二个是找出系统存在的性能瓶颈，内存泄漏，耗时的sql等等。

**测试工具**

性能测试工具有很多，这些工具主要分为两类，第一类是针对网页的测试分析工具，比如LoaderRunner(很强大，但是收费)，Dynatrace(有免费版本，可录制网页操作)，WebPageTest(在线测试工具，网址：http://www.webpagetest.org/)。第二类是针对后端服务程序进行性能监控测试以及结果分析的工具，对于java写的后端来说，比较有代表性的工具是jmeter，jvisualvm，jprofiler。

Jprofiler

1. 安装与配置

服务器端

1. 从<http://www.ej-technologies.com/download/jprofiler/files>上下载linux版本的jprofiler\_linux\_9\_1\_1.sh 文件放到linux服务器中用sh命令执行安装。
2. 安装完毕后，在etc/profile中添加：#Jprofiler Conf  
     JPROFILER\_HOME=/opt/jprofiler7/bin/linux-x64  
     export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$JPROFILER\_HOME
3. 如果服务器运行的是tomcat，那就在catalina.sh中添加：  
     #Jprofiler Conf  
     JAVA\_OPTS="$JAVA\_OPTS -agentlib:jprofilerti=port=8849-Xbootclasspath/a:/opt/jprofiler7/bin/agent.jar"，同时在startup.sh中添加：

 #Jprofiler Conf  
  CATALINA\_OPTS="-agentpath:/opt/jprofiler7/bin/linux-x64/libjprofilerti.so=port=8849,nowait$CATALINA\_OPTS"  
export CATALINA\_OPTS

注意：因为我们的项目是springboot项目，所以没有没有tomcat，那就需要在运行portal.jar 的时候加参数。具体如下nohup java -agentlib:jprofilerti=port=8849 -Xbootclasspath/a:/opt/jprofiler9/bin/agent.jar -jar build/libs/com-imaibo-portfolio-1.0.3.jar --spring.profiles.active=qa &

1. 客户端安装略
2. 客户端连接配置  
   1). 运行 JProfiler 。第一次打开会有向导，忽略它。   
   2). 选择 Session->Integration Wizard->New Remote Integratation   
   3). 选择 On a remote computer；Platform of remote computer 选择 Linux x86/AMD 64；Next   
   4). 输入服务器 IP ；Next   
   5). 输入服务器上的 jprofiler 的安装路径，如 /opt/jprofiler9 ；next   
   6). 选择服务器的 JDK 环境，这里是：Sun，1.8.0，hotspot；next   
   7). 输入端口：这里是默认值 8849；next   
   8). 选择启动模式：这里选第一种 wait for a connection from the jprofiler GUI；next

各项参数

Telemetries：遥感勘测

Overview:各项参数的预览，包括内存，记录的对象，记录的吞吐量,垃圾回收，加载的类，线程，CPU使用情况

Live memory: class/class instance的相关信息。 例如对象的个数，大小，对象创建的方法执行栈，对象创建的热点。

All Objects:所有对象的实例数，大小

Recorded Objects:被记录的对象实例数，大小

Allocation call tree:分配内存的树形结构。

Allocation Hot Spots:内存分配热点

Class Tracker:类的追踪器，具体要追踪的类要从All Objects 或者 Recorded Objects里添加。

Heap walker:堆遍历器，从Live memory的 Objects里面选择显示，可以查看分配情况，最大的对象，关联

CPU views: CPU消耗的分布及时间(cpu时间或者运行时间); 方法的执行图; 方法的执行统计(最大，最小，平均运行时间等)

Hot Spots:每个方法执行的多少次，平均一次多长时间，总共花了多少时间。

Call Graph:调用的关系图表，可以从Call Tree 或者Hot Spots里选择一个方法进行显示。

Method Statistics: 方法统计

Call Tracer: 调用跟踪器

JavaScript XHR:追踪JavaScript的XHR调用，要安装chrome插件

Threads:对线程的监控

Thread History:历史线程的监控

Thread Monitor:线程监控器

Thread Dumps：废弃线程

Montitors & locks：监控器和锁

CurrentLocking Graph:当前锁的图表

Current Monitors：当前的监控器

Locking History Graph：锁的历史图表

Monitors History:历史监控器

Monitor Usage Statistics:监控器的使用统计

Databases:数据库的监控，可以看到那些操作最耗时，触发了多少次，总共多少时间，平均一次多少时间。

JDBC

JPA/Hibernate

MongoDB

Cassandra

HBase

WebPageTest

http://www.webpagetest.org/