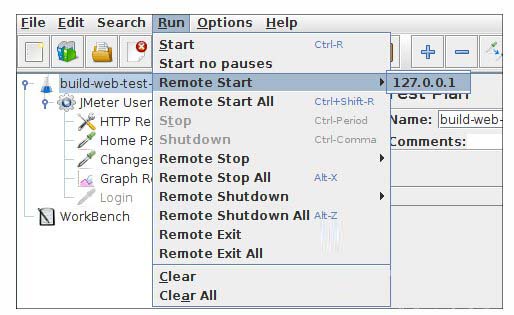
实上，你的JMeter客户端机器是不能表现出完美的压力请求，来模拟足够多的用户或由于网络限制去向服务施加压力，一种解决方法是通过一个JMeter去控制多个/远程JMeter。通过运行JMeter的远程模式，你可以通过多个低配制的电脑复制测试计划用于模拟一个大数据量的压力并发在服务器上。一个JMeter客户端实例可以控制多个远程JMeter实例，并收集所有数据信息。特点如下:  
　　· 保存测试样例在本机电脑  
　　· 通过一台机器来管理多台JMeter测试机  
　　· 不需要复制测试计划到每一台JMeter测试机-JMeter客户端会发送至所有的服务器  
　　提示: 相同的测试计划将会被运行在所有的服务器中。JMeter不会在服务器间分发压力，每一个均会完整执行测试计划。因此，如果你设置了1000个线程并有6个JMeter服务，你将会完成6000线程注入。  
　　然而，远程模式比独立运行相同数量的无UI测试使用更多的资源。如果许多服务实例被使用，JMeter客户端会超负载，直到达到网络链接的上限。通过使用 剥离模式 可以得到提升(见下文)，但你可以一直查看是否你的JMeter是否超负载。  
　　请注意，当你可以在你的应用程序服务器上执行JMeter测试时，你需要知道有一件事实将有会额外的消耗在你的应用服务器中并会影响你的测试结果。推荐的方式时，将JMeter与你的应用服务配制在同一网断。这将最小化由于网络引起影响的性能测试结果。  
**Step 0: 配制节点**  
　　确保所有的节点(客户端和服务):  
　　· 运行相同的JMeter版本  
　　· 在所有系统中使用相同的Java版本。使用不同的Java版本也可正常工作，但不鼓励这样做。  
　　如果在测试过程中使用任何数据文件， 注意这些将不会被通过客户端发送给服务器，因此请确保这些信息可被所有的服务所获取 。如果需要，你可以通过在每个服务中的 user.properties 或 system.properties 来定义不同的属性。当服务启启动时，这些属性将会被使用并可以应用到测试计划中来影响测试操作(如，链接不同的远程服务)。或者，在测试的数据文件中使用不同的数据(如，每个服务使用唯一的id，把这些数据分到不同的数据文件中)  
**Step 1: 启动服务**  
　　使用远程模式运行JMeter，在所有的机器中启动JMeter服务组件通过运行 JMETER\_HOME/bin/jmeter-server (unix) 或 JMETER\_HOME/bin/jmeter-server.bat (windows)脚本。  
　　注意，在每一个节点中仅应该有一个JMeter服务，除非有不同的 RMI 在使用。  
　　自JMeter2.3.1后，JMeter应用服务自行启动 RMI 的服务注册;不需要再单独启 RMI 。为了改变上面的方式，可以定义JMeter的属性  
　　server.rmi.create=false  
　　在服务的系统中。  
　　通常， RMI 为JMeter服务引擎使用动态端口。这样会引起防火墙问题，找到属性名 server.rmi.localport 来控制你的端口号。如果这个是非0，服务器将会使用它来做为本地的端口号。  
**Step 2: 在客户端属性文件中添加服务端IP地址**  
　　在控制JMeter机器中修改属性文件。在 JMETER\_HOME/bin/jmeter.properties 中，找出属性名 remote\_hosts ，添加你运行JMeter的机器IP。多个地址通过逗号分隔。  
　　注意，你可以使用命令行选项 -R 来替代远程主机来使用。有相同的功能，像使用 -r 和 -Jremote\_hosts={serverlist} 。比如:  
　　jmeter-Rhost1，127.0.0.1，host2  
　　如果你定义JMeter的属性 server.exitaftertest=true ，服务器在运行独立测试后将会退出。查看 -X 标记的使用(下文有说明)。  
　**Step 3a: 使用JMeter GUI客户端检查配制**  
　　现在你已经准备控制JMeter客户端。在Windows电脑中，开启客户端通过 bin/jmeter.bat 。在UNIX电脑中，使用脚本 bin/jmeter 。你将注意到 Run 菜单包括两个字菜单: Remote Start 和 Remote Stop (查看下图)。这些菜单包括你发送属性配制文件。使用 Remote Start 和 Remote Stop 代替正常JMeter的 Start 和 Stop 。



**Step 3b: 通过非GUI启动JMeter**  
　　GUI模式应该仅用来在调试模式，作为一种更好的改变，你应该通过非GUI(命令行)模式启动远程测试。命令行如下:  
　　jmeter -n-tscript.jmx -r  
　　或  
　　jmeter -n-tscript.jmx -R server1，server2，...  
　　其它有用的参数:  
　　-Gproperty=value  
　　在所有服务器中定义一个属性(可以出现多次)  
　　-X  
　　测试结束后，退出远程模式  
　　第一个例子将会启动远程测试无论在JMeter属性 remote\_hosts 是否定义;第二个例子从服务列表定义 remote\_hosts 并在远程服务中启动测试。当所有的远程服务停止后，命令行客户端将会退出。  
**手工操作**  
　　在一些场景下，JMeter服务脚本将无法工作(你所使用的操作系统平台不支持JMeter运行开发)。下面是怎样开始JMeter服务通过一种手机的方法:  
　　Step 1a: 启动RMI注册  
　　自JMeter2.3.1版本后，RMI注册通过JMeter服务启动，因此在平常场景下这部分不会生效。为了解决上面的方法，定义JMeter属性 server.rmi.create=false 在服务存放系统中并按下面的操作继续。  
　　JMeter使用 Remote Method Invocation(RMI) 当作远程通讯工具。因此，你需要运行 JDK 自带的RMI注册应用程序(被命名为， rmiregistry )，位置在 bin 目录中。在运行 rmiregistry 之前，确保下面的 jar 包在你的 系统环境变量 中:  
　　· JMETER\_HOME/lib/ext/ApacheJMeter\_core.jar  
　　· JMETER\_HOME/lib/jorphan.jar  
　　· JMETER\_HOME/lib/logkit-2.0.jar  
　　rmiregistry 应用程序需要读取一些JMeter类。运行 rmiregistry 不需要参数。默认监听的端口号是 1099 .  
**Step 1b: 启动JMeter服务**  
　　一旦 RMI 注册应用程序已经运行，启动JMeter服务。使用 -s 参数作为JMeter的启动参数。  
　　第2/3步还是保持一致。  
　　提示  
　　JMeter/RMI需要从客户端至服务器的连接。将会使用你选择的端口号，默认是 1099 .  
　　JMeter/RMI也需要一个回传的链接，用于返回样例结果从服务器端到客户端。这个会使用一个较大数字的端口号。  
　　通过JMeter的 jmeter.properties 的属性 client.rmi.localport 来控制端口号。如果没有防火墙和其它网络过滤在JMeter的客户端和服务器端，你需要确保他们创建的链接是可通的。如果有必要，使用监听软件查看通信被生成。  
　　如果你使用 Suse Linux 系统，这些信息将会有帮助。默认的安装将会启动防火墙。如果这样的话，远程测试将无法工作。下面的方法是 Sergey Ten 提供的。  
　　如果你看到链接被拒绝，通过下面的选项打开调试:  
　　rmiregistry -J-Dsun.rmi.log.debug=true   
　　-J-Dsun.rmi.server.exceptionTrace=true   
　　-J-Dsun.rmi.loader.logLevel=verbose   
　　-J-Dsun.rmi.dgc.logLevel=verbose   
　　-J-Dsun.rmi.transport.logLevel=verbose   
　　-J-Dsun.rmi.transport.tcp.logLevel=verbose   
　　在JMeter2.3.1版本之后，RMI注册被服务启动;但选项仍可在命令行中工作。如: jmeter -s -Dsun.rmi.loader.logLevel=verbose (如，删除 -J 前缀)。改变属性可以在 system.properties 属性文件中定义。  
　　解决问题的方案是删除回调 127.0.0.1 和 127.0.0.2 从 /etc/hosts 。发生了什么引起， jmeter-server 不能链接 rmiregistry 如果 127.0.0.2 回调不可用。使用下面的设置修复:  
**替换**  
　　dirname $0`/jmeter-s"$@"  
　　用  
　　HOST="-Djava.rmi.server.hostname=[computer\_name][computer\_domain]   
　　-Djava.security.policy=`dirname $0`/[policy\_file]"   
　　`dirname $0`/jmeter $HOST -s "$@"  
　　同时创建协议文件，并添加 电脑名 电脑域 行在 /etc/hosts 。  
　　为了更好的支持RMI通讯通道的 SSH 协议在远程测试中，自JMeter2.6版本后:  
　　· 一个新的属性 client.rmi.localport 可以设置来控制RMI端口通过使用 RemoteSampleListenerImpl  
　　· 为了支持RMI使用SSH通信协议作为一个远程终端，在本机使用一个端口，可在Java的系统属性 java.rmi.server.hostname 参数中说明出来的回调接口现在允许调用。

**使用不同的端口号**  
　　默认情况下，JMeter使用标准RMI的端口 1099 。是可以改变的。为了达到这个目的，下面所有的都是执行:  
　　· 在服务器上，使用新的端口号来启动 rmiregistry  
　　· 在服务器上，使用服务器定义的端口号来启动 JMeter  
　　· 在客户端上，更新 remote\_hosts 属性包括新的 host:port 设置  
　　自JMeter2.1.1版本后，JMeter服务端脚本提供支持变更端口。比如，假设你想使用 1664 ( 1099 已被占用)  
　　在Windows电脑中(命令行)  
　　C:JMETER>SETSERVER\_PORT=1664  
　　C:JMETER> JMETER-SERVER [other options]  
　　在Unix电脑中  
　　$SERVER\_PORT=1664jmeter-server [otheroptions]  
　　[环境变量使用大写字母]  
　　在上面两种情况下，在特殊的端口使用脚本启动 rmiregistry ，接着通过已经定义的 server\_port 属性在服务端启动JMeter。  
　　可选端口号将会被输出在服务器的 jmeter.log 文件中( rmiregistry 不会创建日志文件)。  
　　使用不同的测试样例  
　　在测试计划中，监听器发送他们的结果至默认的JMeter客户端。通过情况下，样例结果将会在生成的时候同时返回。这个将会影响服务器的最大吞吐量;样例结果在线程继续时必须返回所有的结果。有一些JMeter属性可以改变这个功能。  
**mode**  
　　样例发送模式-在2.9版本默认采用是 StrippedBatch。这个将会被发送至客户端结点。  
　　· Standard : 在生成样例后即发送结果  
　　· Hold : 保存样例至数组中直到运行结束。这个将会使用大量的内存在服务器端，不推荐使用。  
　　· DiskStore : 存储样例在磁盘文件中( java.io.temp 文件夹下)直到运行结束。JVM退出时，这个文件将会被删除。  
　　· StrippedDiskStore : 从返回数据中删除成功的数据，并使用 DiskStore 发送  
　　· Batch : 发送保存的样例当数量( num\_sample\_threshold )或时间( time\_threshold )达到一个阀值，在这个阀点时将同步发送样例。这个值可以在服务器的下列属性中配制:  
　　  · num\_sample\_threshold : 样例累积数，默认 100  
　　  · time\_threshold : 时间值，默认是 60000毫秒 = 60秒  
　　异步模式的使用，在下面。  
　　· Statistical : 发送概要样例当数量或时间达到阀值。样例将被归整通过线程组名和样例标签。下面是可以归整的属性:  
　　elapsed time  
　　latency  
　　bytes  
　　sample count  
　　error count  
　　· Stripped : 从返回数据中删除成功的样例  
　　· StrippedBatch : 从返回数据中删除成功的样例，并使用 Batch 发送  
　　· Asynch : 样例被临时存储在本地队列中。一个单独的工作线程发送样例。这样就不需要测试线程等待结果发送至客户端。然而，如果线程的创建速度比发送速度快，消息队列将会最终被填满，样例线程会被堵塞直到样例被从队列中清理完全。这个模式比较适用于平滑的样例测试。消息队列的大小可以在服务器端调节通过JMeter的属性 asynch.batch.queue.size (默认 100 )。  
　　· StrippedAsynch : 从返回数据中删除成功的样例，并使用 Asynch 发送  
　　· Custom implementation : 配制模式参数至你的自定义样例类名中。这个必须继承接口 SampleSender 并且有一个独立参数类型为 RemoteSampleListener 的构造方法。  
　　Stripped`模式会包括`responseData`流，因此一些元素前置`responseData`将能正常工作。  
　　这不是真正的问题，将会有更有效的方法来增加这个特性。  
　　下面的属性应用于 Batch 和 Statistical 模式:  
　　num\_sample\_threshold设置的样例数(默认为 100 )  
　　time\_threshold设置时间值(默认 60秒 )  
**处理节点启动失败**  
　　针对大数量级的测试，有一种情况是部分远程服务会不可用或无法启动。比如，当你使用自动化脚本来定位大量的云机器并使用作为并发机器，一些请求机器也许会失败因为一些云的问题。自JMeter2.13后，有一些新的属性来控制这种行为。  
　　首先，你应该尝试重试初始化在那些仅是因为延时启动。为了确保重试，你应该设置 client.tries 属性来设置总的尝试次数。默认情况下仅会尝试一次。为了控制重试延迟，设置 client.retries\_delay 属性来控制在等待尝试的时间间隔(毫秒)。  
　　最后，你应该继续运行使用这些并发机器测试，并成功初始化且跳过失败的节点。为了使这个可用，设置 client.continue\_on\_fail=true 属性。