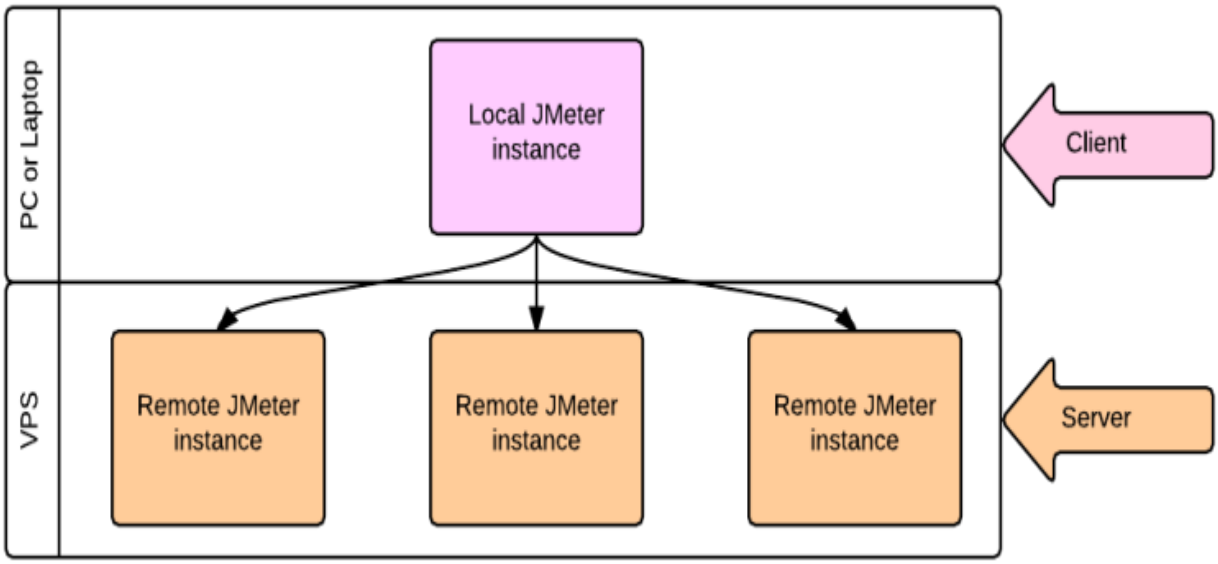
A STUDY ON JMETER AND REMOTE SERVERS

In my previous post I discussed why you might want to run your own servers for load & performance testing. In this article I will elaborate a bit on how to setup your machines. The Apache Jmeter pages of course have an explanation on how to setup remote tests. What I have heard from colleagues and what I experienced myself, is that that explanation is not always as clear and concise as may be preferred. This is my attempt at giving a readable explanation on how to setup JMeter with remote instances.

Some basic terminology

I will use the terms Local and Remote to identify the different sides of the configuration needed to get things working.

- Local is to be read as your workstation, e.g. the machine you use to build your JMeter scripts.
- Remote should be considered any machine that will be running a (headless) JMeter instance and will help generate load on an object.



How to setup JMeter locally to work with remote machines

Configuring JMeter to work with your own (or your customers) load generating servers is relatively easy, however it is not as straightforward as one might hope and expect.

Client Configuration

Firstly we need to add the fully qualified domainnames or IP addresses of the servers to JMeter. These names, or addresses will, after starting JMeter show up in the “Run” menu of Jmeter under the “Remote start” item.

In order to do so take your favorite file-editor and open up “jmeter.properties” on your own, local machine and edit the following section:

```
#---------------------------------------------------------------------------
--

# Remote hosts and RMI configuration

#---------------------------------------------------------------------------
--

# Remote Hosts - comma delimited

remote_hosts=<INSERT YOUR OWN IP ADDRESSES OR NAMES, AS A COMMA SEPARATED
LIST>

client.rmi.localport=7000
```

The two items just edited are the settings for the remote hosts, so the IP addresses of the servers you want to use within your grid. This can become a relatively long list of addresses, since you can technically add as many as you desire. When looking at my personal setup I have several instances of the remote_hosts line, where only one is active at any given point in time. Each customer that has it’s own remote machines, will receive a separate line, which is used when working for that particular customer. This is in order for me to make sure I do not accidentally start a test from servers that I should not be using or which are simply not available at the time. The latter mistake is not as damaging as the first, however I consider it annoying because JMeter will hang for a fairly long time attempting to connect to the remote machines.

The second line defines the port on which the local JMeter instance will listen for the Remote Method Invocation or RMI connections from the servers.

Server configuration

RMI ports

In the JMeter properties file (jmeter.properties) you will have to find the following settings and adjust them to your needs. I have chosen port 7000 for the client RMI port and 60000 for the server RMI port since that was a port that was open within most of my customer firewalls and not linked to anything else. Technically these ports can be left to the defaults filled in in the standard jmeter.properties. Keeping the default value for the client RMI port however will often lead to unpredictability due to firewalls being locked on loads of ports.

```
# Parameter that controls the RMI port used by the
RemoteSampleListenerImpl (The Controler)

# Default value is 0 which means port is randomly assigned

# You may need to open Firewall port on the Controller machine

client.rmi.localport=7000

# To use a specific port for the JMeter server engine, define
# the following property before starting the server:
server.rmi.localport=60000
```

Sample senders

Having set the correct ports to ensure all data gets sent back to the local instance for reporting in your Listeners is step one. The next step is ensuring the data is sent back to your local instance in a way that works best for you. The settings for this can be found in jmeter.properties as well, the part you are looking for is the “Remote batching configuration”

```
#---------------------------------------------------------------------------
--

# Remote batching configuration
```

```
#-------------------------------------------------------------------------
--
```

This part of the configuration may be a bit more sensitive and difficult to get right. There are a lot of different settings to play with depending on the mode you choose. My preferred setting is this:

```
mode=Standard
```

This mode sends samples synchronously as soon as they are generated, thus giving you a constant update on the progress of the test.

In order to save bandwidth though, it may also be preferred to store all samples in memory until the end of the run. I prefer to not do this, since this can lead to loss of data is the JMeter instance crashes due to memory issues, but if you want to preserve bandwidth it is a good option as well.

```
mode=DiskStore
```

Connect the dots: Portforwarding

In order to easily connect with my remote servers and have them ready for testing I have created a simple set of shell scripts. They look something like this:

```
ssh -R 7000:127.0.0.1:7000 USER@SERVER.NAME.COM <<EOF
cd apache-jmeter-2.11/bin/
./jmeter-server
EOF
```

So, what is hapening here?

```
ssh -R 7000:127.0.0.1:7000 USER@SERVER.NAME.COM <<EOF
```

This command will start an SSH reverse tunnel, which connects the remote port 7000 to the local port 7000.

This effectively makes it possible for the server to send data back directly to your local JMeter instance, thus ensuring you can see the results of your tests on your own local machine while the test is running.

Adding the *<<EOF* to ensure the script, once logged in will keep executing commands until it reaches **E**nd **O**f **F**ile, where the script exits.

```
cd apache-jmeter-2.11/bin/
```

Once logged in to the remote machine, change directory to where the JMeter executable is installed. In this particular example I assume the JMeter instance used is not a default installed binary but a custom version. If you use a standard JMeter install on a default Linux distribution (i.e. Ubuntu) this step can be omitted.

```
./jmeter-server

EOF
```

Start the JMeter Server instance and signal to the script that the end of file has been reached and no more commands need to be sent to the remote server. Instead of executing “jmeter-server” it is also possible to start the server with the basic JMeter startup script by adding the -s flag, thus resulting in the following command:

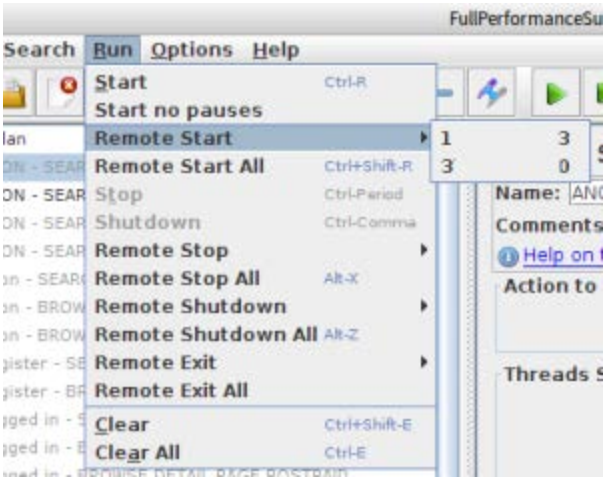
```
jmeter -s

EOF
```

You can add these steps for all servers in separate shell scripts or of course make it one big setup script that by default will start all Jmeter-servers within your grid.

Now that the servers have been started it is time to run a test to see if this actually worked.

Starting the test

Starting a test with the remote machines is relatively easy once all of the above has been done. In JMeter hit the Run menu and select either the machine you would like to use or select the Remote Start All to just start all machines.



A very important thing to keep in mind in running tests on a set of remote machines is the following (as shown on the Jmeter Usermanual page as well):

Note: The same test plan is run by all the servers. JMeter does not distribute the load between servers, each runs the full test plan.

In other words, if you set your testplan to have 500 threads and you run this test on 4 servers you will effectively start not 500 threads spread out over those servers, you will start 500 threads *per server* thus resulting in 4000 threads.

If you do not want to start the test from the GUI there is also the possibility of starting remote tests from commandline. You can choose to either start on all the servers you have configured:

```
jmeter -n -t script.jmx -r
```

Or you can be picky and specify the machines you want to start the tests on:

```
jmeter -n -t script.jmx -R server1,server2
```

In my tests I quite often use CSV files to feed data to my tests, for example to feed usersnames and passwords, search terms etc. In order for these CSV files to be picked up on the server side as well, they need to be uploaded to the server. Where JMeter will send the testscript to the server, it does not send any data-files to the servers.

Source : http://martijndevrieze.net/2014/12/02/jmeter-and-remote-servers-a-tutorial/