SpeedTester

DistributeMe SpeedTester

From time to time you might want to know how your software will behave before actually using it. That's the place where DistributeMe SpeedTester comes into the game.

The SpeedTester is a very simple Service implementation with two clients, one that is making RemoteCalls in SingleThreaded mode and one - in multithreaded.

How To Use

1. Download the attached speedtester.tgz file - for example (wget --no-check-certificate https://confluence.opensource.anotheria.net/download/attachments/13763039/speedtester.tgz)
2. Unpack it in a directory of your choice with

   ```
   tar -xzf speedtester.tgz
   ```

   command. Note, on mac Safari unpacks gziped files automatically.
3. Change into speedtester directory.
4. Start the server with

   ```
   ./startServer.sh
   ```

   . You will see some debug output:

   ```
   log4j:WARN No appenders could be found for logger (net.anotheria.anoprise.metafactory.
   ConfigurableResolver).
   log4j:WARN Please initialize the log4j system properly.
   log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
   skipping registration for org_distributeme_speedtester_TestingService
   Server org_distributeme_speedtester_TestingService is up and ready.
   ```

   Once you've seen last line, the server is ready (usually in the same second you started it).
5. Open another terminal window, and change into the same directory (speedtester).
6. Now start the client:

   ```
   ./startRemoteClient.sh localhost 9249
   ```

   . You will see something like this (output from my MBP):

```
Testing with rmi://org_distributeme_speedtester_TestingService.xxx@localhost:9249
Testing instance of org.distributeme.speedtester.generated.RemoteTestingServiceStub@c0b76fa
connection established
Testing echo calls
        1 - echo calls, Performed 1000 in 311, 0.311 ms per call - OK
Testing inbound packets, size 1000
        2 - inbound packets with size 1000, Performed 1000 in 349, 0.349 ms per call - OK
Testing inbound packets, size 10000
        3 - inbound packets with size 10000, Performed 1000 in 368, 0.368 ms per call - OK
Testing inbound packets, size 50000
        4 - inbound packets with size 50000, Performed 1000 in 659, 0.659 ms per call - ACCEPTABLE
Testing outbound packets, size 1000
        5 - outbound packets with size 1000, Performed 1000 in 348, 0.348 ms per call - OK
Testing outbound packets, size 10000
        6 - outbound packets with size 10000, Performed 1000 in 402, 0.402 ms per call - OK
Testing outbound packets, size 50000
        7 - outbound packets with size 50000, Performed 1000 in 948, 0.948 ms per call - ACCEPTABLE
Testing in and out bound packets, size 1000
        8 - in-and-out bound packets with size 1000, Performed 1000 in 312, 0.312 ms per call - OK
Testing in and out bound packets, size 10000
        9 - in-and-out bound packets with size 10000, Performed 1000 in 451, 0.451 ms per call - OK
Testing in and out bound packets, size 50000
        10 - in-and-out bound packets with size 50000, Performed 1000 in 772, 0.772 ms per call -
ACCEPTABLE
=========================
Finished Test In 4935 ServerSide: 4930
=========================
1 - echo calls, Performed 1000 in 311, 0.311 ms per call - OK
2 - inbound packets with size 1000, Performed 1000 in 349, 0.349 ms per call - OK
3 - inbound packets with size 10000, Performed 1000 in 368, 0.368 ms per call - OK
4 - inbound packets with size 50000, Performed 1000 in 659, 0.659 ms per call - ACCEPTABLE
5 - outbound packets with size 1000, Performed 1000 in 348, 0.348 ms per call - OK
6 - outbound packets with size 10000, Performed 1000 in 402, 0.402 ms per call - OK
7 - outbound packets with size 50000, Performed 1000 in 948, 0.948 ms per call - ACCEPTABLE
8 - in-and-out bound packets with size 1000, Performed 1000 in 312, 0.312 ms per call - OK
9 - in-and-out bound packets with size 10000, Performed 1000 in 451, 0.451 ms per call - OK
10 - in-and-out bound packets with size 50000, Performed 1000 in 772, 0.772 ms per call - ACCEPTABLE
```

7. Congrats you just executed your first speed test, with one thread. Now proceed with multithreaded test.
8. Start multithreaded test:

```
./startMultiThreadedRemoteClient.sh 10 localhost 9249
```

and watch the output:

```
Testing with rmi://org_distributeme_speedtester_TestingService.xxx@localhost:9249
Testing instance of org.distributeme.speedtester.generated.RemoteTestingServiceStub@44b01d43
connection established
-- testing echo calls with 10 threads
        1 - echo calls, Performed 10000 in 816, 0.0816 ms per call - SUPERB
-- testing inbound packets with size 1000 with 10 threads
        2 - inbound packets with size 1000, Performed 10000 in 774, 0.0774 ms per call - SUPERB
-- testing inbound packets with size 10000 with 10 threads
        3 - inbound packets with size 10000, Performed 10000 in 1094, 0.1094 ms per call - TOP
-- testing inbound packets with size 50000 with 10 threads
        4 - inbound packets with size 50000, Performed 10000 in 4307, 0.4307 ms per call - OK
-- testing outbound packets with size 1000 with 10 threads
        5 - outbound packets with size 1000, Performed 10000 in 858, 0.0858 ms per call - SUPERB
-- testing outbound packets with size 10000 with 10 threads
        6 - outbound packets with size 10000, Performed 10000 in 2718, 0.2718 ms per call - TOP
-- testing outbound packets with size 50000 with 10 threads
        7 - outbound packets with size 50000, Performed 10000 in 14588, 1.4588 ms per call - LOUSY
-- testing in-and-out bound packets with size 1000 with 10 threads
        8 - in-and-out bound packets with size 1000, Performed 10000 in 526, 0.0526 ms per call - SUPERB
-- testing in-and-out bound packets with size 10000 with 10 threads
        9 - in-and-out bound packets with size 10000, Performed 10000 in 1151, 0.1151 ms per call - TOP
-- testing in-and-out bound packets with size 50000 with 10 threads
        10 - in-and-out bound packets with size 50000, Performed 10000 in 4546, 0.4546 ms per call - OK
=========================
Finished Test In 31414 ServerSide: 31413 with 10
=========================
1 - echo calls, Performed 10000 in 816, 0.0816 ms per call - SUPERB
2 - inbound packets with size 1000, Performed 10000 in 774, 0.0774 ms per call - SUPERB
3 - inbound packets with size 10000, Performed 10000 in 1094, 0.1094 ms per call - TOP
4 - inbound packets with size 50000, Performed 10000 in 4307, 0.4307 ms per call - OK
5 - outbound packets with size 1000, Performed 10000 in 858, 0.0858 ms per call - SUPERB
6 - outbound packets with size 10000, Performed 10000 in 2718, 0.2718 ms per call - TOP
7 - outbound packets with size 50000, Performed 10000 in 14588, 1.4588 ms per call - LOUSY
8 - in-and-out bound packets with size 1000, Performed 10000 in 526, 0.0526 ms per call - SUPERB
9 - in-and-out bound packets with size 10000, Performed 10000 in 1151, 0.1151 ms per call - TOP
10 - in-and-out bound packets with size 50000, Performed 10000 in 4546, 0.4546 ms per call - OK
```

9. Congrats, you're done! Now write down your results and send them to me

What are the options

startRemoteClient.sh

This script understands two parameters, the target host (the machine the server is running on) and the port. The default value for port is 9249 (configurable in the *startServer.sh*.
Examples:

- ./startRemoteClient.sh - starts a new client, which connects to the server at port 9249 on localhost.
- ./startRemoteClient.sh 192.168.1.1 - starts a new client, which connects to the server at port 9249 on 192.168.1.1.
- ./startRemoteClient.sh 192.168.1.1 9350 - starts a new client, which connects to the server at port 9350 on 192.168.1.1. Note that you have to configure the server properly to run it under 9350 (see startServer.sh)

startMultiThreadedRemoteClient.sh

This script accepts same parameters as the above, but its first parameter has to be the number of threads.
Examples:
./startMultiThreadedRemoteClient.sh - starts a new client that connects to localhost, port 9249 with 10 threads.
./startMultiThreadedRemoteClient.sh 20 - starts a new client that connects to localhost, port 9249 with 20 threads.
./startMultiThreadedRemoteClient.sh 5 localhost - - starts a new client that connects to localhost, port 9249 with 5 threads.
./startMultiThreadedRemoteClient.sh 50 192.168.1.1 9350 - - starts a new client that connects to 192.168.1.1, port 9350 with 50 threads.

How To Read

The test performs different calls between two Java VMs and measures the roundtrip duration. Naturally the shortest time is best, however, the performance of a single call is not that important as the throughput. The multithreaded test usually performs better, because it uses the cpu wait times to make multiple calls in parallel.

The test consists of 10 use cases, first it performs simple echo calls, without data load. Those calls should be well below 0.1 ms.
Than it performs in, out and in-out data calls with 1000, 10.000 and 50.000 bytes. In - means that the data is sent to server, out - retrieved from server and in-out send and received from server.

The times should be always below one second (expect maybe 50K times) calling localhost from localhost. Echo should be always below 0.1 ms if testing in local network.

Example times.

Just some examples, actual results may differ (be better I hope .-)

HE to HE, 10 Threads

==========
Finished Test In 169907 ServerSide: 169904 with 10
==========
1 - echo calls, Performed 10000 in 776, 0.0776 ms per call - SUPERB
2 - inbound packets with size 1000, Performed 10000 in 1481, 0.1481 ms per call - TOP
3 - inbound packets with size 10000, Performed 10000 in 2816, 0.2816 ms per call - TOP
4 - inbound packets with size 50000, Performed 10000 in 15667, 1.5667 ms per call - LOUSY
5 - outbound packets with size 1000, Performed 10000 in 1522, 0.1522 ms per call - TOP
6 - outbound packets with size 10000, Performed 10000 in 20300, 2.03 ms per call - LOUSY
7 - outbound packets with size 50000, Performed 10000 in 109851, 10.9851 ms per call - DEAD HORSE
8 - in-and-out bound packets with size 1000, Performed 10000 in 627, 0.0627 ms per call - SUPERB
9 - in-and-out bound packets with size 10000, Performed 10000 in 2712, 0.2712 ms per call - TOP
10 - in-and-out bound packets with size 50000, Performed 10000 in 13985, 1.3985 ms per call - LOUSY

Hetzner to Hetzner

==========
Finished Test In 159836 ServerSide: 159833 with 10
==========
1 - echo calls, Performed 10000 in 730, 0.073 ms per call - SUPERB
2 - inbound packets with size 1000, Performed 10000 in 991, 0.0991 ms per call - SUPERB
3 - inbound packets with size 10000, Performed 10000 in 8627, 0.8627 ms per call - ACCEPTABLE
4 - inbound packets with size 50000, Performed 10000 in 42955, 4.2955 ms per call - LOUSY
5 - outbound packets with size 1000, Performed 10000 in 964, 0.0964 ms per call - SUPERB
6 - outbound packets with size 10000, Performed 10000 in 8605, 0.8605 ms per call - ACCEPTABLE
7 - outbound packets with size 50000, Performed 10000 in 42938, 4.2938 ms per call - LOUSY
8 - in-and-out bound packets with size 1000, Performed 10000 in 950, 0.095 ms per call - SUPERB
9 - in-and-out bound packets with size 10000, Performed 10000 in 8852, 0.8852 ms per call - ACCEPTABLE
10 - in-and-out bound packets with size 50000, Performed 10000 in 44187, 4.4187 ms per call - LOUSY

MBP via Wlan, T-Com DSL to HE, 10 Threads

1 - echo calls, Performed 10000 in 27997, 2.7997 ms per call - LOUSY

MBP via Wlan, T-Com DSL to HE, 50 Threads

1 - echo calls, Performed 50000 in 42663, 0.85326 ms per call - ACCEPTABLE

MBP via Wlan, T-Com DSL to Hetzner, 50 Threads

1 - echo calls, Performed 50000 in 43445, 0.8689 ms per call - ACCEPTABLE

Hetzner to HE

==========
Finished Test In 259241 ServerSide: 259223 with 10
==========
1 - echo calls, Performed 10000 in 8548, 0.8548 ms per call - ACCEPTABLE
2 - inbound packets with size 1000, Performed 10000 in 8583, 0.8583 ms per call - ACCEPTABLE
3 - inbound packets with size 10000, Performed 10000 in 9550, 0.955 ms per call - ACCEPTABLE
4 - inbound packets with size 50000, Performed 10000 in 42983, 4.2983 ms per call - LOUSY
5 - outbound packets with size 1000, Performed 10000 in 8714, 0.8714 ms per call - ACCEPTABLE
6 - outbound packets with size 10000, Performed 10000 in 10763, 1.0763 ms per call - LOUSY
7 - outbound packets with size 50000, Performed 10000 in 67890, 6.789 ms per call - DEAD HORSE
8 - in-and-out bound packets with size 1000, Performed 10000 in 8831, 0.8831 ms per call - ACCEPTABLE
9 - in-and-out bound packets with size 10000, Performed 10000 in 10488, 1.0488 ms per call - LOUSY
10 - in-and-out bound packets with size 50000, Performed 10000 in 82838, 8.2838 ms per call - DEAD HORSE

How Many Threads

Difficult question, but here are some measures, taken between two workstations in a corporate network.
10 - 1 - echo calls, Performed 10000 in 730, 0.073 ms per call - SUPERB
20 - 1 - echo calls, Performed 20000 in 1111, 0.05555 ms per call - SUPERB
30 - 1 - echo calls, Performed 30000 in 1358, 0.04526666666666666 ms per call - SUPERB
50 - 1 - echo calls, Performed 50000 in 2015, 0.0403 ms per call - SUPERB
100 - 1 - echo calls, Performed 100000 in 3203, 0.03203 ms per call - SUPERB
200 - 1 - echo calls, Performed 200000 in 5837, 0.029185 ms per call - SUPERB
500 - 1 - echo calls, Performed 500000 in 13736, 0.027472 ms per call - SUPERB

FAQ

Why is the default port 9249?

Because the default port range for services is 9250-9299 and 9249 will not interfere with any running services on the same machine.

What is the best suitable amount of threads?

We don't now, try it out 😅

Is Speedtester a good example on how to use DistributeMe?

NO! It's usage isn't typical for DistributeMe and shouldn't be taken as example. You can find a much better example here:

```
svn://svn.anotheria.net/opensource/distributeme-examples/trunk/helloworld
```

.
However, you can use Speedtester as an example for Peer-2-Peer client/service configuration.

Where can I find the sources?

```
http://svn.anotheria.net/opensource/distributeme-examples/trunk/dime-speedtester
```