How can I find the response time of a HTTP request through a Socket

Asked 16 years, 3 months ago Modified 3 years, 8 months ago

Viewed 108k times

21

I'm using a Java socket, connected to a server. If I send a HEADER http request, how can I measure the response time from the server? Must I use a provided java timer, or is there an easier way?

I'm looking for a short answer, I don't want to use other protocols etc. Obviously do I neither want to have a solution that ties my application to a specific OS. Please people, IN-CODE solutions only.

java http

Share

Improve this question

Follow

edited Sep 16, 2008 at 3:17

community wiki
3 revs
Martin Andersson

For debugging purposes, or for use by your program?
– owenmarshall Sep 16, 2008 at 2:50

7 Answers

Sorted by: Highest score (default)

21

```
curl -s -w "%{time_total}\n" -o /dev/null
http://server:3000
```

Share Improve this answer

Follow

answered Sep 21, 2012 at 8:08

community wiki
A B

14

I would say it depends on what exact interval you are trying measure, the amount of time from the last byte of the request that you send until the first byte of the response that you receive? Or until the entire response is received? Or are you trying to measure the server-side time only?

If you're trying to measure the server side processing time only, you're going to have a difficult time factoring out the amount of time spent in network transit for your request to arrive and the response to return. Otherwise, since you're managing the request yourself through a Socket, you can measure the elapsed time between any

two moments by checking the System timer and computing the difference. For example:

```
public void sendHttpRequest(byte[] requestData, Socket
    long startTime = System.nanoTime();
    writeYourRequestData(connection.getOutputStream(),
    byte[] responseData = readYourResponseData(connect
    long elapsedTime = System.nanoTime() - startTime;
    System.out.println("Total elapsed http request/res
nanoseconds: " + elapsedTime);
}
```

This code would measure the time from when you begin writing out your request to when you finish receiving the response, and print the result (assuming you have your specific read/write methods implemented).

Share Improve this answer Follow

edited Jan 22, 2020 at 16:47

answered Sep 16, 2008 at 3:44

David L

44.6k ● 11 ● 64 ● 62

1 where can i add this function??in my web service?
– Aditya Vyas-Lakhan Feb 24, 2015 at 5:03

1 Elapsed time should be measure by using `System.nanoTime()`, see stackoverflow.com/a/1776053/1839228 – Franz Becker Jan 22, 2020 at 12:02

@FranzBecker You're quite right. Now that Java 1.4 is well past end of life, it's safe to assume that System.nanoTime() is always available. – David L Jan 22, 2020 at 16:47

12

You can use time and curl and time on the command-line. The -I argument for curl instructs it to only request the header.

```
time curl -I 'http://server:3000'
```

Share Improve this answer

Follow

answered Sep 16, 2008 at 3:11

hoyhoy
6,351 ● 7 ● 40 ● 36

9

Something like this might do the trick

```
import java.io.IOException;

import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.HttpMethod;
import org.apache.commons.httpclient.URIException;
import org.apache.commons.httpclient.methods.HeadMetho
import org.apache.commons.lang.time.StopWatch;
//import org.apache.commons.lang3.time.StopWatch
```

```
public class Main {

    public static void main(String[] args) throws URIE
        StopWatch watch = new StopWatch();
        HttpClient client = new HttpClient();
        HttpMethod method = new HeadMethod("http://sta

        try {
            watch.start();
            client.executeMethod(method);
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            watch.stop();
        }

        System.out.println(String.format("%s %s %d: %s
method.getURI(), method.getStatusCode(), watch.toStrin

    }
}
```

```
HEAD http://stackoverflow.com/ 200: 0:00:00.404
```

Share Improve this answer

Follow

edited Apr 9, 2021 at 7:16

Pang

10.1k ● 146 ● 85 ● 124

answered Sep 16, 2008 at 3:38

Dave Cheney

5,755 ● 2 ● 21 ● 24

3

Maybe I'm missing something, but why don't you just use:

```
// open your connection
long start = System.currentTimeMillis();
// send request, wait for response (the simple socket
```

```
long end = System.currentTimeMillis();
System.out.println("Round trip response time = " + (en
```

Share Improve this answer

Follow

edited Sep 16, 2015 at 22:28

answered Sep 16, 2008 at 3:41

Kevin Day

16.4k ● 8 ● 48 ● 71

0

Use AOP to intercept calls to the socket and measure the response time.

Share Improve this answer

Follow

answered Sep 16, 2008 at 4:54

Adisesha

5,258 ● 1 ● 34 ● 44

0

```
@Aspect
@Profile("performance")
@Component
public class MethodsExecutionPerformance {
    private final Logger logger = LoggerFactory.getLog

    @Pointcut("execution(* it.test.microservice.myServ
    public void serviceMethods() {
    }

    @Around("serviceMethods()")
    public Object monitorPerformance(ProceedingJoinPoi
throws Throwable {
        StopWatch stopWatch = new StopWatch(getClass()
```

```
        stopWatch.start();
        Object output = proceedingJoinPoint.proceed();
        stopWatch.stop();
        logger.info("Method execution time\n{}", stopW
        return output;
    }
}
```

In this way, you can calculate the real response time of your service independent of network speed.

Share Improve this answer Follow

answered Jan 24, 2019 at 13:49

community wiki
Soodabeh Neirizi