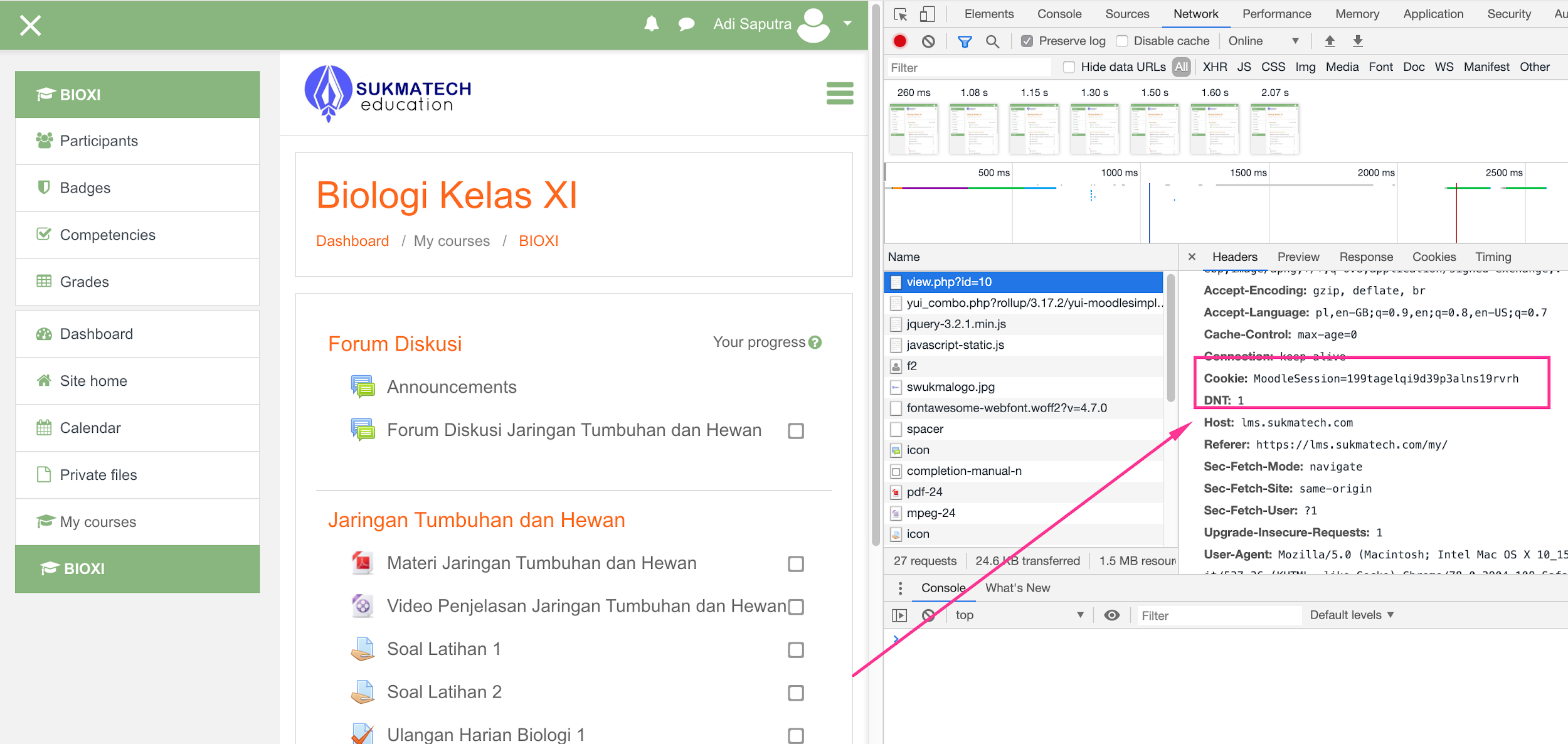
**hey** (<https://github.com/rakyll/hey>) is a very simple tool similar to ApacheBench so you can only test one URL and type of request. Of course you can run two or more instances to simulate traffic on different URLs or by different users.

In this case if you want to test your site you need to authorize your user.

1. Run Chrome
2. Enter tested page
3. Log in
4. Run Developer mode
   1. View > Developer > Developer Tools
5. Click on first link on this page
6. Look i Developer panel send headers to server and find Cookie



Now you see that this page is using Moodle ;)

Now you can test any link on your page as logged user using hey - just copy this header to **hey**:

|  |
| --- |
| hey -c 2 -n 10 -m GET -H "Cookie: MoodleSession=199tagelqi9d39p3alns19rvrh" "https://lms.sukmatech.com/course/view.php?id=10" |

MoodleSession=gnsip8dv64eiiq824cfe3v17o0

hey -c 2 -n 10 -m GET -H "Cookie: MoodleSession=gnsip8dv64eiiq824cfe3v17o0" "https://lms.sukmatech.com/course/view.php?id=10"

And result:

|  |
| --- |
|  |

As you see we have made 10 requests and all of them get in response HTTP code 200 (that's mean OK).

If you want to post a message on a forum you can do the same thing as in the GET method.

First prepare file with POST request body:

My **forum\_post.txt** file:

|  |
| --- |
| timestart=0&timeend=0&course=10&forum=0&discussion=5&parent=5&groupid=0&edit=0&reply=5&sesskey=eyiTvvF8EF&\_qf\_\_mod\_forum\_post\_form=1&mform\_isexpanded\_id\_general=1&mform\_isexpanded\_id\_tagshdr=1&subject=Re%3A+Forum+Diskusi+Bersama&message%5Btext%5D=freelancer.com+test%3Cbr%3E&message%5Bformat%5D=1&message%5Bitemid%5D=911671341&discussionsubscribe=1&attachments=312023137&tags=\_qf\_\_force\_multiselect\_submission&tags%5B%5D=test&submitbutton=Post+to+forum |

No change in hey method to POST and add one more header “Content-Type”:

|  |
| --- |
| hey -c 2 -n 10 -m POST -D ./forum\_post.txt -H "Cookie: MoodleSession=dlnepc38jdv7hvsdqrm9u7pcjt" -H "Content-Type: application/x-www-form-urlencoded" "https://lms.sukmatech.com/mod/forum/post.php" |

MoodleSession=gnsip8dv64eiiq824cfe3v17o0

hey -c 2 -n 10 -m POST -D /home/faisal\_ssi/forum.txt -H "Cookie: MoodleSession=gnsip8dv64eiiq824cfe3v17o0" -H "Content-Type: application/x-www-form-urlencoded" "https://lms.sukmatech.com/mod/forum/post.php"

Of course send it to the URL as above [**https://lms.sukmatech.com/mod/forum/post.php**](https://lms.sukmatech.com/mod/forum/post.php) .

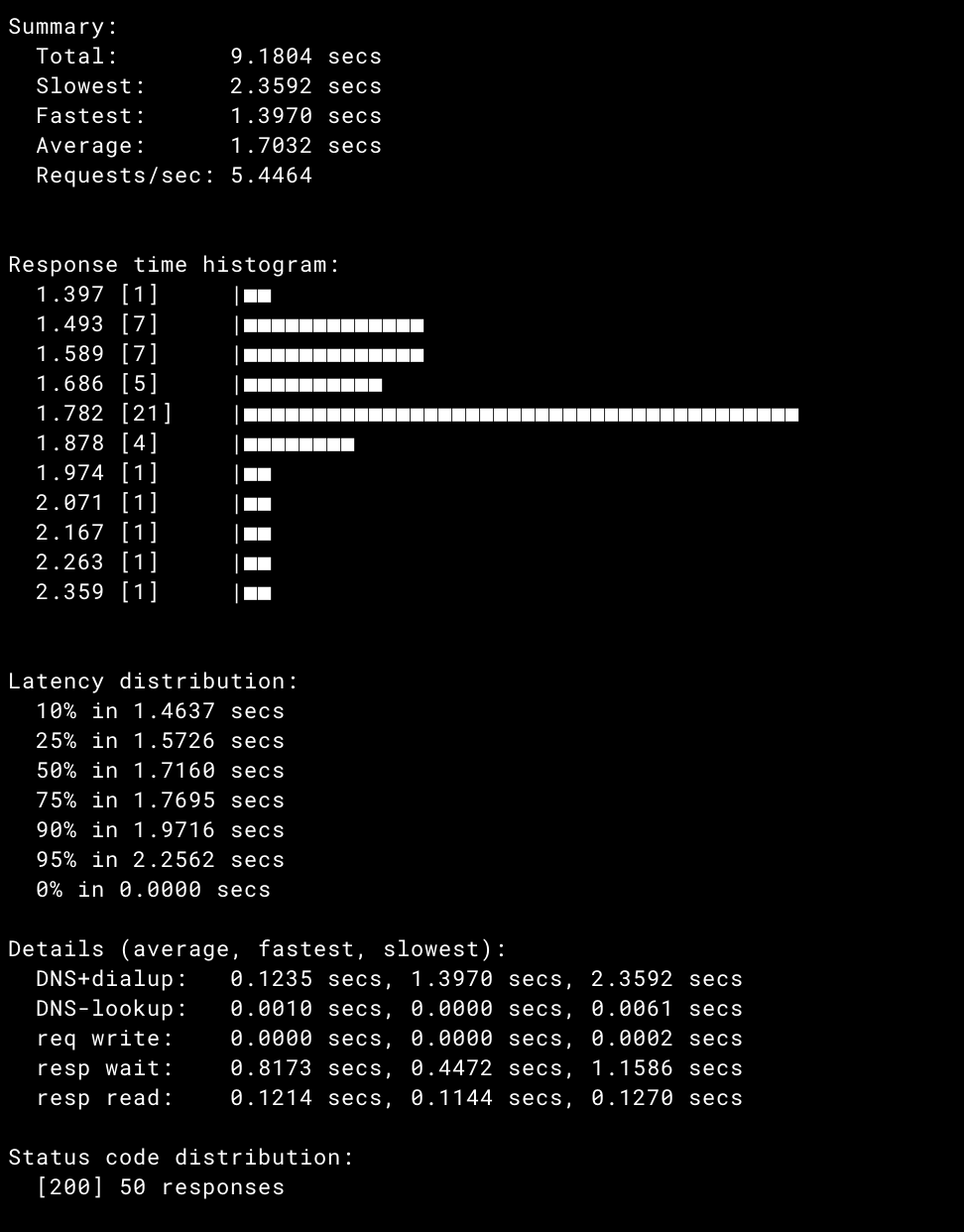
If you want to make a test as 2 users or more just collect Cookie data for each user and run **hey** in parallel for each user separately.

|  |
| --- |
|  |

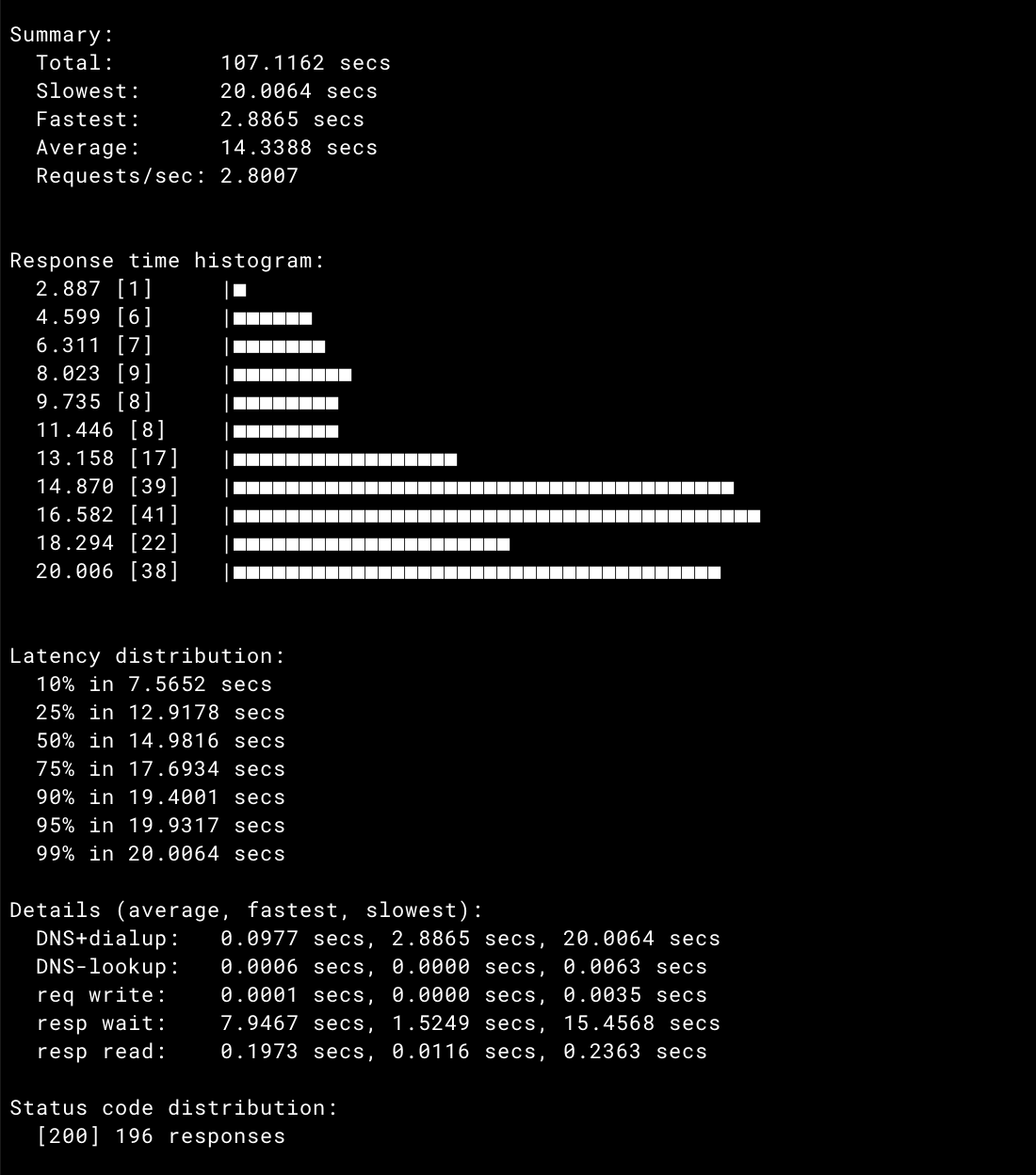
With **Hey** it's the simplest way to test your website. Of course you can use such tools like jMeter, Gatling, Loctus, k6 but you will need to spend much more time learning how to prepare tests using them.

I have made two additional tests.

**10** concurrent workers (users) are making **50** posts on forum:



**50** concurrent workers (users) are making **300** posts on forum in the same time:



The result for **10** users is **very good**, and for **50** it is **satisfactory** (196 correctly served query).

Remember that it will not happen that 50 users will be sending 6 posts at the same time.

Note:

it looks like the average time to complete the request

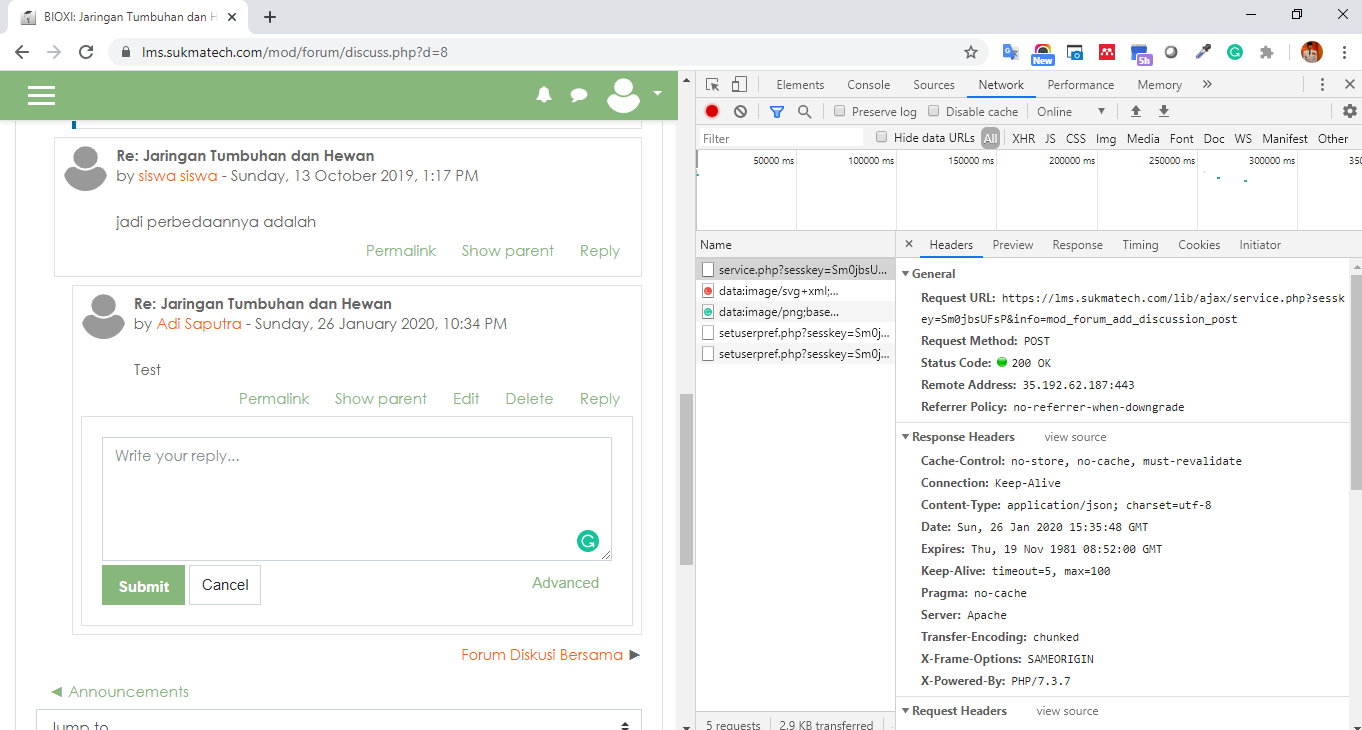
10% o requests was made in less than 7sec

Another tester

<https://k6.io/>

<https://gatling.io/>

[https://loadimpact.com/](https://www.freelancer.com/users/l.php?url=https:%2F%2Floadimpact.com%2F&sig=ea3614139553294136be1df28407eb4048a7bc03b9c9421d5031ec36bbc59514" \t "_blank)



# SIEGE

faisal\_ssi@stress-test:/etc/siege$ siege -c25 -t60s --header="Cookie: MoodleSession=se3uim8mei9op5bpj14808g53a" 'https://lms.sukmatech.com/mod/forum/view.php?id=29'

\*\* SIEGE 4.0.4

\*\* Preparing 25 concurrent users for battle.

The server is now under siege...

Lifting the server siege...

Transactions: 2455 hits

Availability: 100.00 %

Elapsed time: 59.31 secs

Data transferred: 61.09 MB

Response time: 0.59 secs

Transaction rate: 41.39 trans/sec

Throughput: 1.03 MB/sec

Concurrency: 24.47

Successful transactions: 2455

Failed transactions: 0

Longest transaction: 3.92

Shortest transaction: 0.13

TEST POST BERHASIL

**faisal\_ssi@stress-test:/etc/siege$ siege -c25 -t10s --header "Cookie: MoodleSession=se3uim8mei9op5bpj14808g53a"**

**--header "Content-Type: application/x-www-form-urlencoded" 'https://lms.sukmatech.com/mod/forum/post.php POST < payload.txt'**

IS ASCII: TRUE

\*\* SIEGE 4.0.4

\*\* Preparing 25 concurrent users for battle.

The server is now under siege...

Lifting the server siege...

Transactions: 302 hits

Availability: 100.00 %

Elapsed time: 9.64 secs

Data transferred: 7.68 MB

Response time: 0.78 secs

Transaction rate: 31.33 trans/sec

Throughput: 0.80 MB/sec

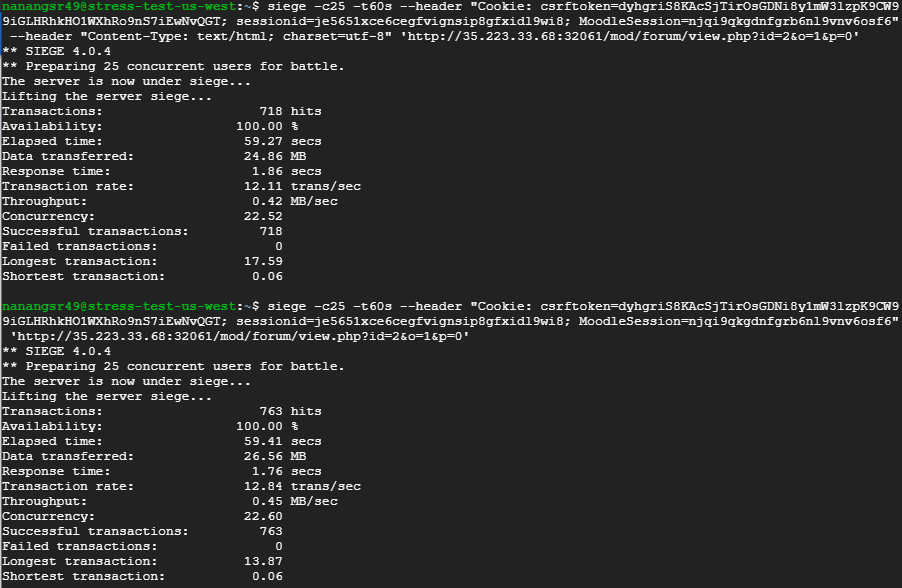
Concurrency: 24.53

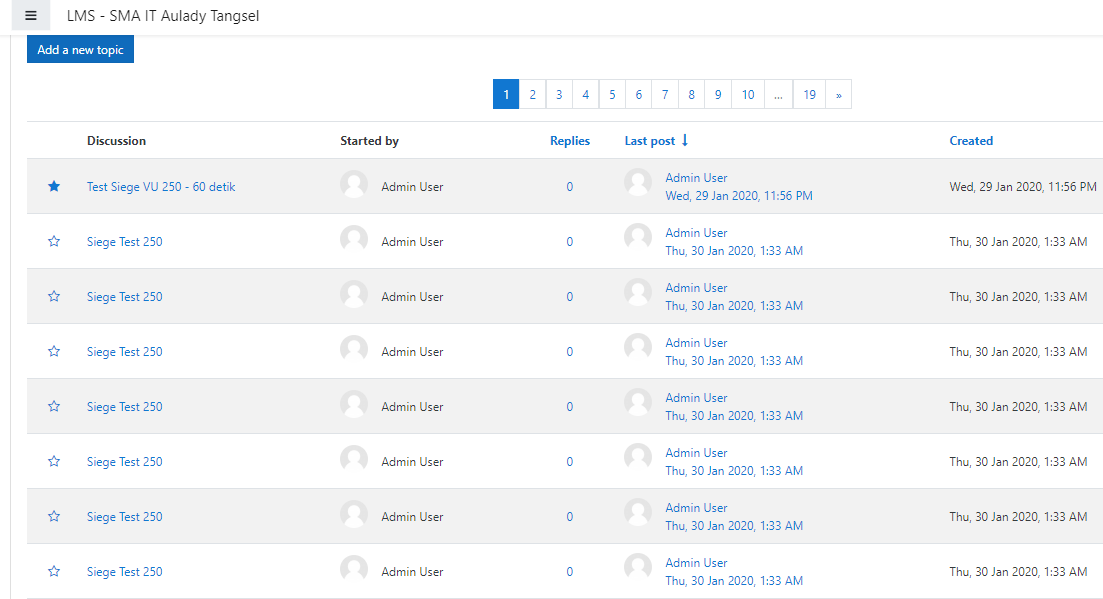
Successful transactions: 305

Failed transactions: 0

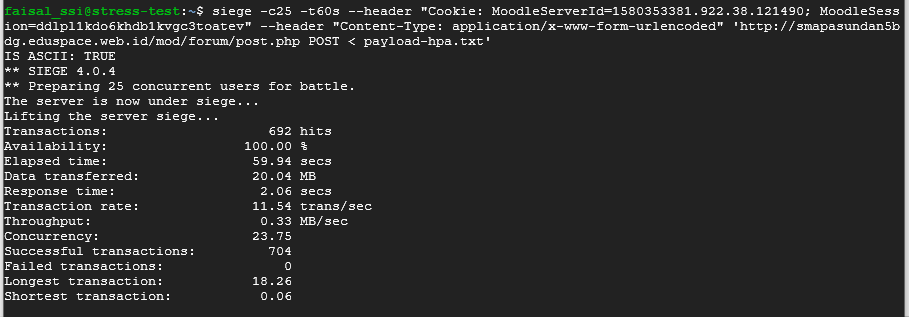
Longest transaction: 4.43

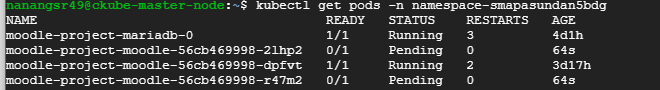
Shortest transaction: 0.13

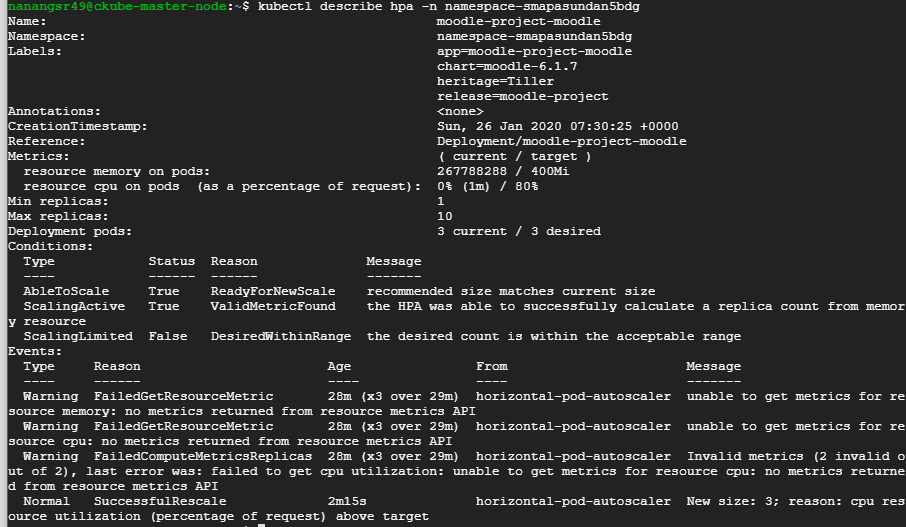




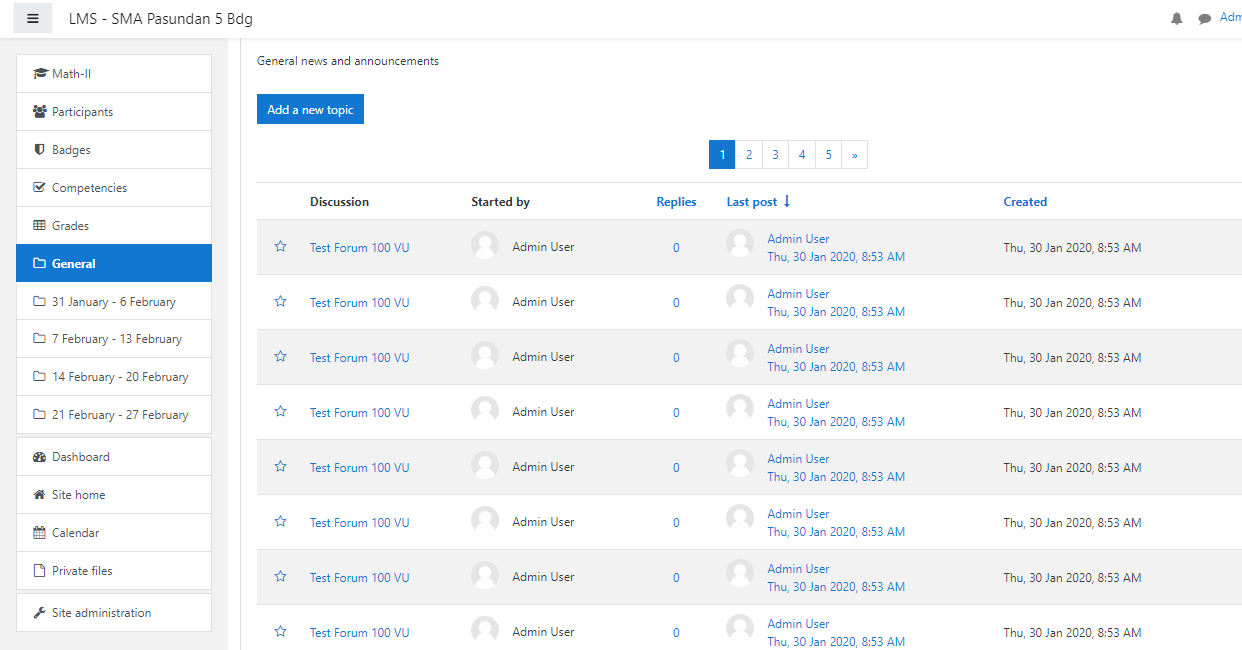
TEST HPA



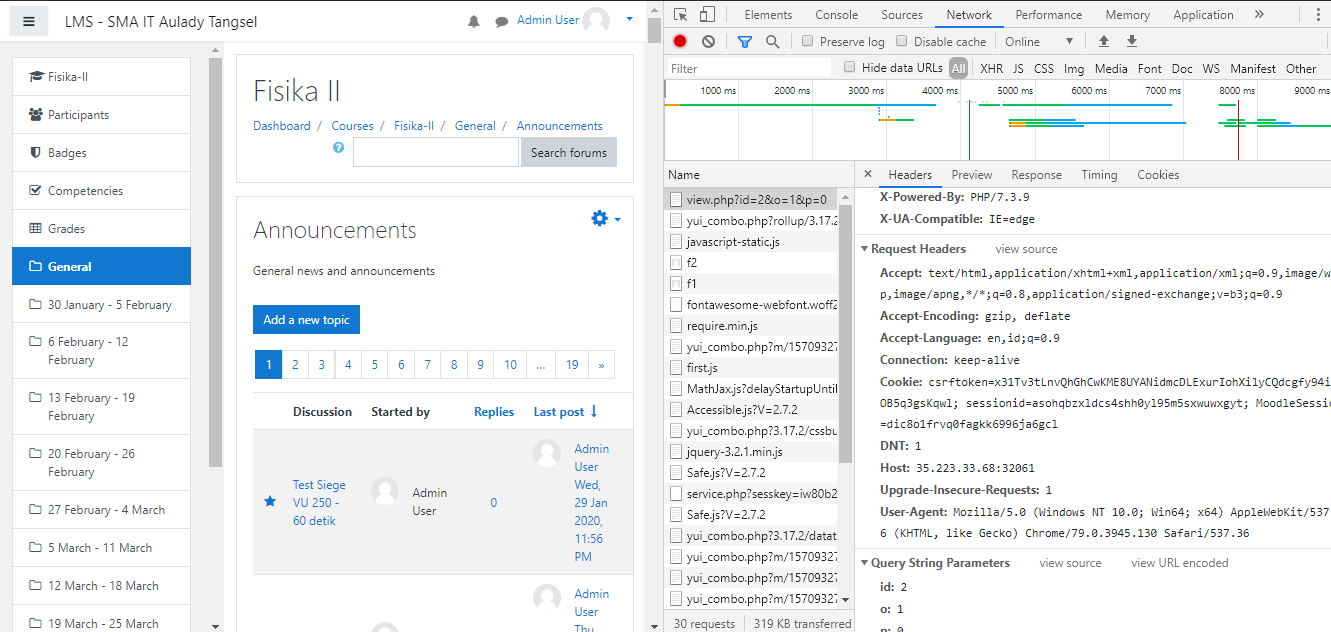




396 baris forum ditambahkan



## Test GET 175 Users



# REF

<https://www.sitepoint.com/web-app-performance-testing-siege-plan-test-learn/> => bagus buat pembahasan

### Interpreting The Results of a Load Test

Now, first of all, my results are not that impressive, as I’m running this on a low powered virtual machine on my local machine, if you’re load testing production sites, you should expect much faster response times.

#### Terminology

**Transactions** is the number of server hits. In the example, 385 transactions.

**Elapsed time** is the duration of the entire siege test. This is measured from the time the user invokes siege until the last simulated user completes its transactions. Shown above, the test took 76.02 seconds to complete.

**Data transferred** is the sum of data transferred to every siege simulated user. It includes the header information as well as content. Because it includes header information, the number reported by siege will be larger then the number reported by the server. In internet mode, which hits random URLs in a configuration file, this number is expected to vary from run to run.

**Response time** is the average time it took to respond to each simulated user’s requests.

**Transaction rate** is the average number of transactions the server was able to handle per second, in a nutshell: transactions divided by elapsed time.

**Throughput** is the average number of bytes transferred every second from the server to all the simulated users.

**Concurrency** is average number of simultaneous connections, a number which rises as server performance decreases.

**Successful transactions** is the number of times the server returned a code less then 400. Accordingly, redirects are considered successful transactions.