Load Test Report

Date: 5/27/2015

Test from: virginia

Query URL: docey.com

Started at: Wed May 27 2015, 02:28:33 -04:00

Finished at: Wed May 27 2015, 02:29:33 -04:00

Test link: https://www.blitz.io/to#/play

Analysis

This rush generated 37,373 successful hits in 60 seconds and we transferred 458.47 MB of data in and out of your app. The average hit rate of 623/second translates to about 53,817,120 hits/day.

The average response time was 338 ms.

You've got bigger problems, though: 6.45% of the users during this rush experienced timeouts or errors!

Response Times | Test Configuration | Other Stats
--- | --- | ---
Slowest: 979 ms | Duration: 60 seconds | Transferred: 5.05MB
Average: 338 ms | Load: 1-2000 users | Received: 453.42MB

Hits

This rush generated 37,373 successful hits. The number of hits includes all the responses listed below. For example, if you only want HTTP 200 OK responses to count as Hits, then you can specify --status 200 in your rush.

<table>
<thead>
<tr>
<th>Code</th>
<th>Type</th>
<th>Description</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>HTTP</td>
<td>OK</td>
<td>37373</td>
</tr>
</tbody>
</table>

HTTP 200 OK 100% (37373)

Errors

The first error happened at 20 seconds into the test when the number of concurrent users was at 665. Errors are usually caused by resource exhaustion issues, like running out of file descriptors or the connection pool size being too small (for SQL databases).

<table>
<thead>
<tr>
<th>Code</th>
<th>Type</th>
<th>Description</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>TCP</td>
<td>Connection reset</td>
<td>256</td>
</tr>
<tr>
<td>23</td>
<td>TCP</td>
<td>Connection timeout</td>
<td>215</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response duration overlimit</td>
<td>4</td>
</tr>
</tbody>
</table>

Timeouts

The first timeout happened at 42.5 seconds into the test when the number of concurrent users was at 1416. Looks like you've been rushing with a timeout of 1000 ms. Timeouts tend to increase with concurrency if you have lock contention of sorts. You might want to think about in-memory caching using redis, memcached or varnish to return stale data for a period of time and asynchronously refresh this data.
The max response time was: **978 ms @ 2000 users**

The max hit rate was: **1,006 hits per second**