Load Test Report

Date: 7/21/2016

Test from: virginia

Query URL: http://2016reviewsignalturbo.a2hosted.com/
Started at: Thu Jul 21 2016, 03:04:57 -04:00
Finished at: Thu Jul 21 2016, 03:05:57 -04:00
Test link: https://www.blitz.io/to#/play

Analysis

This rush generated 590 successful hits in 60 seconds and we transferred 10.30 MB of data in and out of your app. The average hit rate of 10/second translates to about 849,600 hits/day.

The average response time was 92 ms.

You've got bigger problems, though: 97.91% of the users during this rush experienced timeouts or errors!

Response Times

Fastest: 55 ms
Slowest: 167 ms
Average: 92 ms

Test Configuration

Region: virginia
Duration: 60 seconds
Load: 1-1000 users

Other Stats

Avg. Hits: 10/second
Transferred: 2.18 MB
Received: 8.12 MB

HITS

HTTP 200 OK 100% (590)

Errors 96.53% (27255)
Timeouts 1.38% (390)

Hits

This rush generated 590 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>590</td>
</tr>
</tbody>
</table>

Errors

The first error happened at 15 seconds into the test when the number of concurrent users was at 249. 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>13677</td>
</tr>
<tr>
<td>23</td>
<td>TCP</td>
<td>Connection timeout</td>
<td>13578</td>
</tr>
</tbody>
</table>

Timeouts

The first timeout happened at 7.5 seconds into the test when the number of concurrent users was at 124. 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: 167 ms @ 208 users

The max hit rate was: 60 hits per second