Load Test Report

Date: 7/16/2016

Test from: virginia

Query URL: http://reviewsignal2.conetixtest.com/
Started at: Sat Jul 16 2016, 08:25:13 -04:00
Finished at: Sat Jul 16 2016, 08:26:13 -04:00
Test link: https://www.blitz.io/to#/play

Analysis

This rush generated 155 successful hits in 60 seconds and we transferred 5.25 MB of data in and out of your app. The average hit rate of 3/second translates to about 223,200 hits/day.
The average response time was 1,470 ms.
You've got bigger problems, though: 99.50% of the users during this rush experienced timeouts or errors!

Response Times
Fastest: 872 ms
Slowest: 2,184 ms
Average: 1,470 ms

Test Configuration
Region: virginia
Duration: 60 seconds
Load: 1-3000 users
Avg. Hits: 3 /sec
Transferred: 2.21 MB
Received: 3.04 MB

Other Stats
Avg. Hits: 3/sec
Transferred: 2.21 MB
Received: 3.04 MB

Hits
This rush generated 155 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>155</td>
</tr>
</tbody>
</table>

HTTP 200 OK 100% (155)

Errors
The first error happened at 15 seconds into the test when the number of concurrent users was at 746. 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>51</td>
</tr>
<tr>
<td>23</td>
<td>TCP</td>
<td>Connection timeout</td>
<td>15337</td>
</tr>
<tr>
<td>500</td>
<td>HTTP</td>
<td>Internal Server Error</td>
<td>1439</td>
</tr>
</tbody>
</table>

Connection reset 0% (51)
Connection timeout 91% (15337)
HTTP 500 Internal... 9% (1439)

Timeouts
The first timeout happened at 7.5 seconds into the test when the number of concurrent users was at 370. Looks like you've been rushing with a timeout of 2000 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: 2184 ms @ 370 users

The max hit rate was: 36 hits per second