Load Test Report

Date: 7/20/2016

Test from: virginia

Query URL: http://2016liquidwebprofessional.reviewsignal.com/

Started at: Wed Jul 20 2016, 07:18:34 -04:00
Finished at: Wed Jul 20 2016, 07:19:34 -04:00

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

Analysis

This rush generated **81,393** successful hits in **60 seconds** and we transferred **1.43 GB** of data in and out of your app. The average hit rate of **1,357/second** translates to about **117,205,920** hits/day.

The average response time was **80 ms**.

You've got bigger problems, though: **0.07%** of the users during this **rush** experienced timeouts or errors!

### Response Times

- **Fastest**: 76 ms
- **Slowest**: 118 ms
- **Average**: 80 ms

### Test Configuration

- **Region**: virginia
- **Duration**: 60 seconds
- **Load**: 1-3000 users

### Other Stats

- **Avg. Hits**: 1,357/sec
- **Transferred**: 12.60 MB
- **Received**: 1,449.75 MB

### Hits

This rush generated **81,393** 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>81393</td>
</tr>
</tbody>
</table>

### Errors

The first error happened at **47.5 seconds** into the test when the number of concurrent users was at **2373**. 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>23</td>
<td>TCP</td>
<td>Connection timeout</td>
<td>47</td>
</tr>
</tbody>
</table>

### Timeouts

The first timeout happened at **40 seconds** into the test when the number of concurrent users was at **1998**. 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: 117 ms @ 3000 users

The max hit rate was: 2,639 hits per second