LOAD TEST REPORT

DATE: 7/24/2014

TEST FROM: VIRGINIA

Query URL: http://reviewsignal.trypl.com

Started at: Thu Jul 24 2014, 12:51:30 -04:00
Finished at: Thu Jul 24 2014, 12:51:30 -04:00

ANALYSIS

This rush generated 35,547 successful hits in 60 seconds and we transferred 370.77 MB of data in and out of your app. The average hit rate of 592/second translates to about 51,187,680 hits/day.

The average response time was 326 ms.

You've got bigger problems, though: 13.76% of the users during this rush experienced timeouts or errors!

### RESPONSE TIMES

- **Fastest:** 7 ms
- **Slowest:** 752 ms
- **Average:** 326 ms

### TEST CONFIGURATION

- **Region:** VIRGINIA
- **Duration:** 60 seconds
- **Load:** 1-2000 users

### OTHER STATS

- **Avg. Hits:** 592/sec
- **Data Transferred:** 370.77 MB

### HITS

This rush generated 35,547 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>35547</td>
</tr>
</tbody>
</table>
HTTP 200 OK 100% (35547)

ERRORS

The first error happened at **32.5 seconds** into the test when the number of concurrent users was at **1081**. 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>4104</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Response duration overlimit</td>
<td>1</td>
</tr>
</tbody>
</table>

TIMEOUTS

The first timeout happened at **35 seconds** into the test when the number of concurrent users was at **1165**. 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: **752 ms @ 1999 users**

The max hit rate was: **764 hits per second**