Load Test backend

The backend handles, primarily, the upload from the mobile devices of the keys (`/v1/post`); and a small key-exchange prior to this(`/v1/register`). The expected load of these is primarily a function by the number of positive GGD results communicated each day and how many of these patients use the app[1].

The load consists of one `/v1/post` and `/v1/register` call for each of these and roughly 20 times that volume in *decoy* traffic. The register call has just one TEK - to increase the number of transactions/load or overhead relative to the bandwidth.

Therefore the 'worst' expect load is, in a bad situation, 10k patients[2]/day $x(20+1) =$ 0.5 Milion request pairs per day. When spread over the extended working hours and skipping weekends - this yields some 10 to 15 request per second.

Goal

The goal of this test 1) is to confirm that the backend can handle this number several times over and 2) to observe if any thresholds or specific monitoring is required to deal with that load.

Out of scope

GGD portal - login The GGD portal/login has not been load-tested. As 1) GHOR system is not part of the corona app and 2) this system is not consumer facing.

CDN - dist files The CDN has not been load-tested in the context of this backend check.

APEX The apex-page & similar URLs that are not in use have not been tested.

[1] Once most users run the app for more than 14 days - the volume/user is roughly the same - only influenced by epidemiological parameters.

[2] Currently, mid August 2020, around 750 patients/day

Methodology of the test

A high end machine (10 Gbit/second network, directly internet connected, 48 core, 1 Tbyte of memory) with a modern, well tuned Network stack (FreeBSD/12.1) was set up. On this a number of tools, such as 'ab' and the shell scripts used by the corona project where loaded.

A Hardware random generator was connected, providing at least 16x1024 bits of randomness per SSL connection setup/teardown (in addition to the normal Kernel/yarrow entropy).

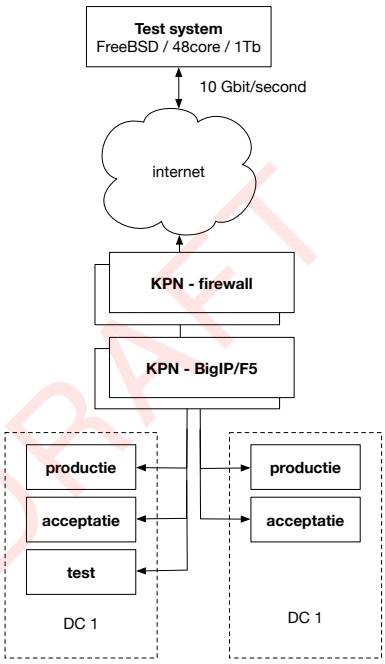

Figure 1: Setup

The were ran against Test, Acceptatie and Production on 2020-07-17 from 13:00 to 14:00 UTC. No special setup of the F5/BigIP, firewalls or other systems was made.

Test

After a verification on the (static, GET) APEX confirmed a sufficiently good network connection (> 300 request/second, > 5 Gbit/Second) - a full register and post session of TEKs request pair was repeatedly carried out on Production, Test and Acceptatie.

Each test was repeated 500 times at 12 concurrency rates - and this was done 3 times on all 3 environment. The averages are shown in figure 2.

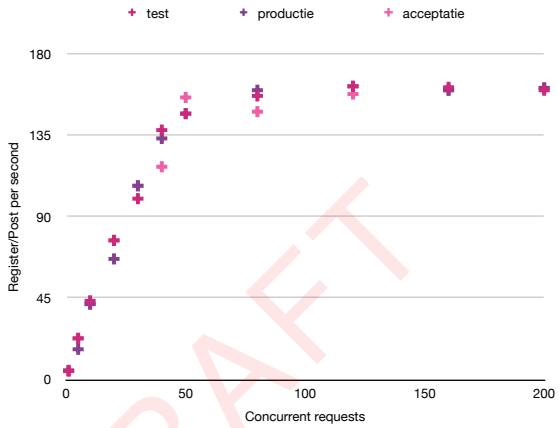

Figure 2: Concurrency versus request-pair/second (rate)

During these tests - operations observed an appreciable increase of load - but still at the very bottom range of capacity. Bandwidth generally around 8-12 Mbit/second sustained; peaking to 40 Mbit/second at times.

Apex anomaly

Between 13:08 and 13:12 *exactly* 20% of the GET calls on '/' failed (at moderate load; < 100 requests/second). This issue went away - and could not be reproduced. An extensive check with the KPN network staff [redacted] did not find anything in the logs, nor did tests (with much higher loads) with KPN staff checking as it ran managed to reproduce it.

Findings

It was found that all 3 systems (the single strand test system and the double strand Acceptance and Production systems) all sustain well over 100 requests/pairs per second. Which is a factor 10 higher than is required.

Document revision history and status

version	changes
1.00	first version (2020-08-11).
1.01	minor typos. BW cross checked. (2020-08-12).