Technical Challenge: URL Shortener

Author: Victor Martinez <vcrmartinez@gmail.com>
Created at: Aug 22, 2023
Github Tracking: https://github.com/victormartinez/urlshortener

Summary

This document briefly outlines the architectural decisions that build up the technical challenge of developing a URL Shortener service that supports 1M RPM peaks of traffic.

Architecture

The designed architecture for the solution is illustrated by Figure 1, which evidences two high-level layers: API and Database.

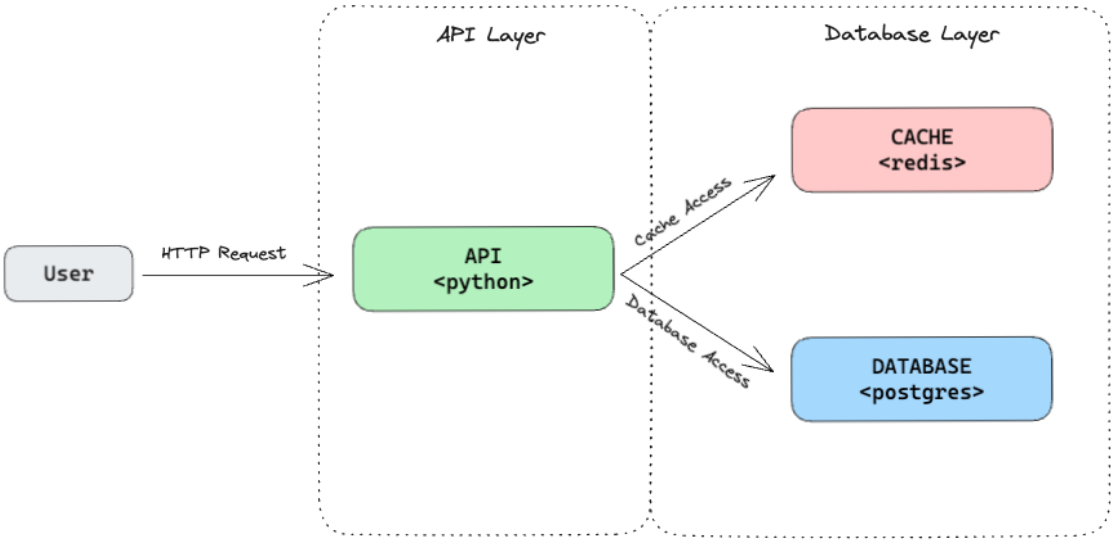

Figure 1: Overview of architectural components.

The user makes a request to API layer built in Python, which can access Cache and/or Database depending on the logic pointed as follows:

- Creating a shortened URL populates both cache and database;
- Updating a shortened URL flushes the record from cache;
- Reading requests makes the API layer follow the steps below:
    - At first, it accesses the cache layer in order to find the destination URL;
    - If cache misses, then API accesses the database.

Key Takeaways

- Database connection is only established if cache misses; since this application is read-intensive and cache hit is expected, the application avoids database connection overhead.
- The usage of a cache layer speeds up URL resolution and avoids latency coming from database trip

Load Test

In order to ensure the load requirements, this project configures a container topology illustrated by Figure 2. The docker-compose configuration spans a NGINX container that works as a Load Balancer for five instances of the application.

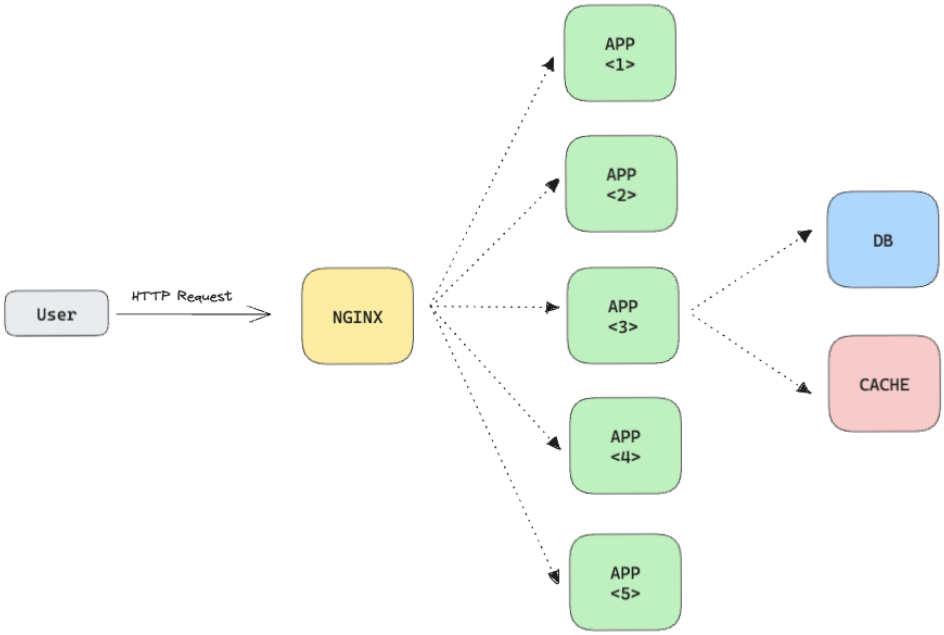

Figure 2: Container architecture regarding load test.

The configuration script specifies a constant arrival rate of 10000 RPM (above the requirements) with 500 pre-allocated virtual users. The load test result is illustrated by Figure 3.

```
         /\      |‾‾| /‾‾/   /‾‾/
    /\  /  \     |  |/  /   /  /
   /  \/    \    |     (   /   ‾‾\
  /          \   |  |\  \ |  (‾)  |
 / __________ \  |__| \__\ \_____/ .io

  execution: local
     script: tests/load/script.js
     output: -

  scenarios: (100.00%) 3 scenarios, 1501 max VUs, 1m55s max duration (incl. graceful stop):
           * warm_up: 1 looping VUs for 5s (gracefulStop: 30s)
           * rump_up_load: Up to 500 looping VUs for 20s over 1 stages (gracefulRampDown: 30s, startTime: 5s, gracefulStop: 30s)
           * constant_request_rate: 166.67 iterations/s for 1m0s (maxVUs: 500-1000, startTime: 25s, gracefulStop: 30s)

INFO[0000] CODE: fkuoDE                                  source=console

running (1m25.2s), 0000/1001 VUs, 46358 complete and 0 interrupted iterations
warm_up               ✓ [======================================] 1 VUs          5s
rump_up_load          ✓ [======================================] 000/500 VUs    20s
constant_request_rate ✓ [======================================] 0000/0500 VUs  1m0s  166.67 iters/s

     ✓ is status success

     █ setup

     checks.........................: 100.00% ✓ 46358      ✗ 0
     data_received..................: 9.0 MB  106 kB/s
     data_sent......................: 4.7 MB  55 kB/s
     http_req_blocked...............: avg=10.75µs  min=0s       max=4.94ms   p(90)=8µs      p(95)=11µs     count=46359
     http_req_connecting............: avg=5.81µs   min=0s       max=1.84ms   p(90)=0s       p(95)=0s       count=46359
     http_req_duration..............: avg=109.88ms min=942µs    max=840.98ms p(90)=316.26ms p(95)=404.99ms count=46359
       { expected_response:true }...: avg=109.88ms min=942µs    max=840.98ms p(90)=316.26ms p(95)=404.99ms count=46359
     ✓ { status:200 }...............: avg=109.88ms min=942µs    max=840.98ms p(90)=316.27ms p(95)=404.99ms count=46358
     ✓ { status:201 }...............: avg=165.12ms min=165.12ms max=165.12ms p(90)=165.12ms p(95)=165.12ms count=1
     ✓ { status:202 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:307 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:308 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:400 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:401 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:403 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:404 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:409 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:500 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:501 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:502 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:503 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     ✓ { status:504 }...............: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=0
     http_req_failed................: 0.00%   ✓ 0          ✗ 46359
     http_req_receiving.............: avg=34.32µs  min=8µs      max=4.39ms   p(90)=59µs     p(95)=79µs     count=46359
     http_req_sending...............: avg=15.14µs  min=2µs      max=3.96ms   p(90)=36µs     p(95)=41µs     count=46359
     http_req_tls_handshaking.......: avg=0s       min=0s       max=0s       p(90)=0s       p(95)=0s       count=46359
     http_req_waiting...............: avg=109.83ms min=909µs    max=840.96ms p(90)=316.23ms p(95)=404.96ms count=46359
     http_reqs......................: 46359   544.262913/s
     iteration_duration.............: avg=109.98ms min=989.83µs max=841.03ms p(90)=316.33ms p(95)=405.03ms count=46359
     iterations.....................: 46358   544.251173/s
     vus............................: 500     min=1        max=500
     vus_max........................: 1001    min=1001     max=1001
```

Figure 3: Load test results.