Numbrs

Numbers URL shortener web service

Ziv Kalderon
Israel
12.20.2017

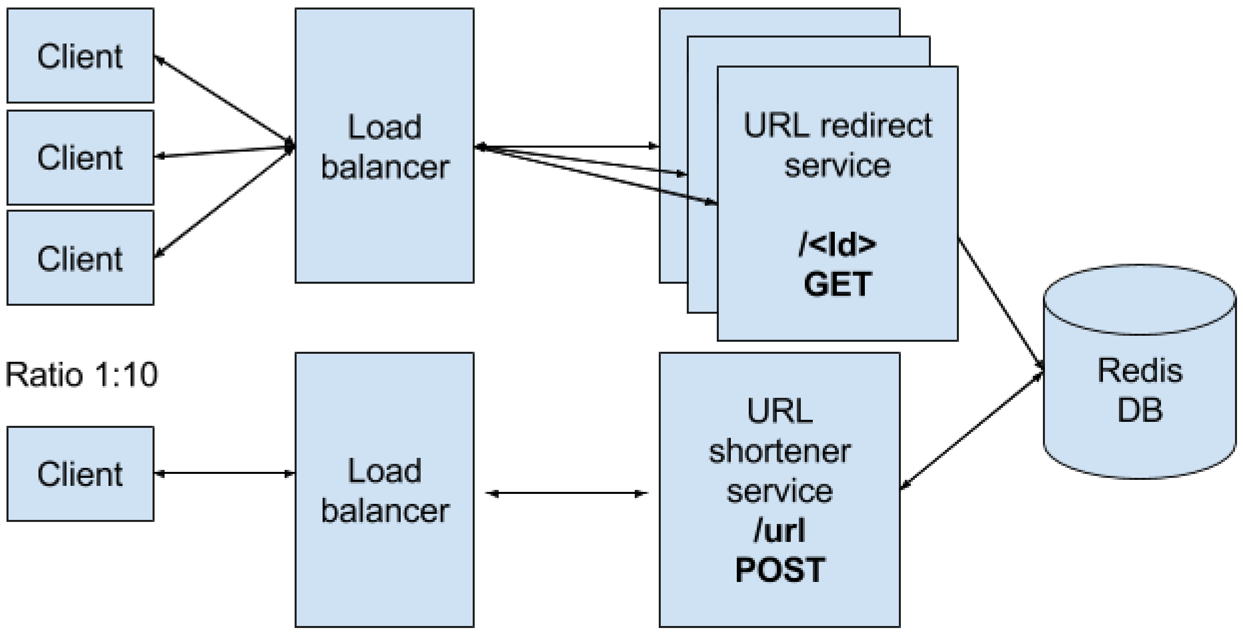
Requirements

- We'll generate consistent shorter and unique id from each URL
- Requests with the shorter URL will redirect to the original link
- High availability of the system - without it we won't be able to redirect requests
    - Scale out design
    - Monitoring
- Redirect requests with minimum latency

High level design

The solution contains two services and one shared DB

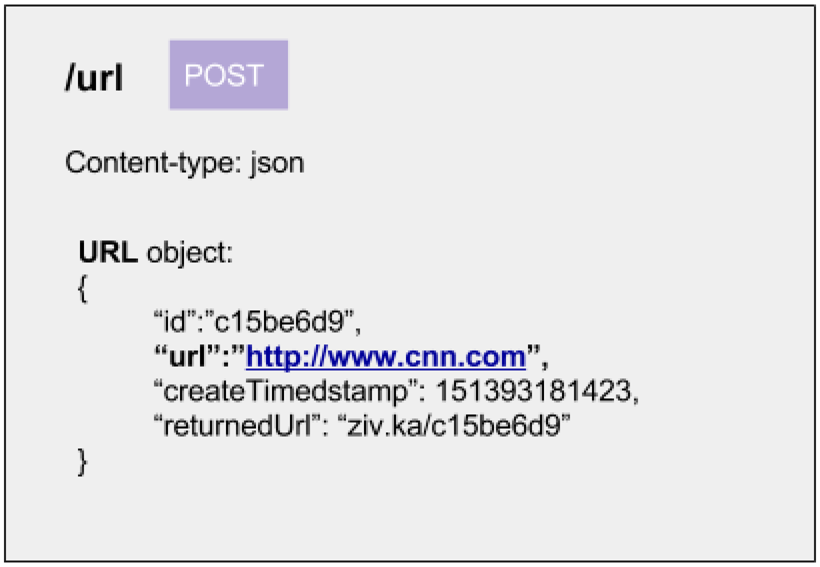
- **Redirect service** - Will support requests with shorter URL. This service will need to support high number of requests (comparing the Shortener service) and be able to be with zero downtime. This service perform **only read** from **DB**
- **Shortener service** - Will support request to short a URL. Each request will get the URL, generate unique hash and return the shorten URL for it. The service **read and write** from the **DB**



- Each service will have load balancer with round robin
- The two services share the same DB which is high available DB and easy to scale out (Support clustering)

APIs

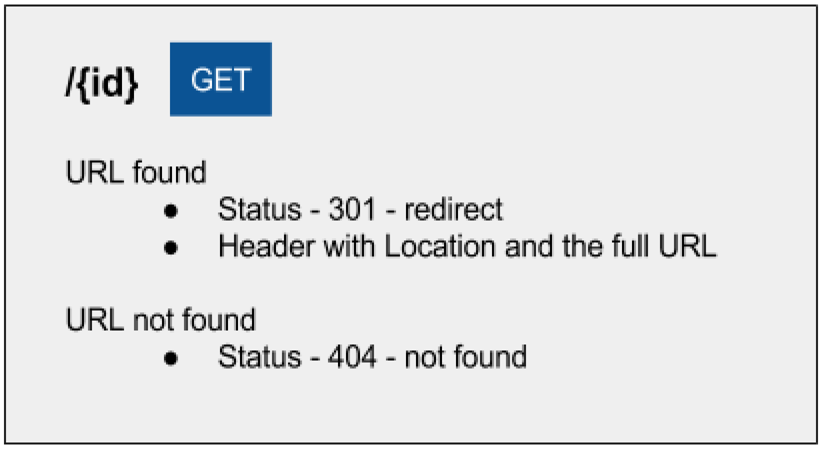
We are supporting two REST Apis



/url POST

Content-type: json

URL object:
```
{
        "id":"c15be6d9",
        "url":"http://www.cnn.com",
        "createTimedstamp": 151393181423,
        "returnedUrl": "ziv.ka/c15be6d9"
}
```



/{id} GET

URL found
- Status - 301 - redirect
- Header with Location and the full URL

URL not found
- Status - 404 - not found

Data Base

The current schema is simple and we currently have only one object. On the other hand we want to support easy scale out and fast reading. That's the reason I chose **Redis.** It's a key value DB that use in memory data structure which makes it really fast. It has a solution for high availability (Redis Sentinel) and scale out options (Redis Cluster). Since it's on memory DB there are two kind of persistence solutions

- RDB - Backup the DB on scheduling. Good for backups and disaster recovery
- AOF - Save every write. Background task

Hashing algorithm

Our use for this algorithm is non-cryptographic, and on the other hand we want to have good performance in uniqueness and speed, additionally we don't want it to be too long (it's a shortening URL service).
That's the reason I chose **32-bit murmur3 algorithm** (https://github.com/google/guava/wiki/HashingExplained)**.**
It's simple and fast, has good distribution (Chi-square tests), good Avalanche effect and collision resistance. A nice article about this algorithem (https://www.ctheu.com/2017/08/26/the-murmur3-hash-function--hashtables-bloom-filters-hyperloglog/)

- **Consistency**
  - 32-bit murmur3 is consistent, for every string it will generate the same hash
- **Performance**
  - Comparing to the cryptographic is much faster
- **Collision support**
  - We've added support for an option of hash collision. In case the hashing algorithm generated the same id for a different URL we're trying to add some random char and regenerate the ID (Good results for this algorithm in avalanche test, change in one character will create new hash that it's different in more than 15%). We will keep trying until we've generated a unique id, or reached the max limit of tries.

Scalability

This solution can be done by using the scale infrastructure of the cloud service (e.g. elastic beanstalk in AWS) or by implementing on own with gateway service (e.g. Zuul) and some discovery service (e.g. kubernetes) with scaling triggers

- **Services**
  - We've separated into two different services in order to support the scale of the redirect service much quicker.
  - For each service we'll have a trigger according to the response time. once the average response time is higher than ~100ms (currently locally ~30ms) it will create new instance
  - Also should support scale down in order to save money
  - Minimum number of running instances should be more than one
- **DB**
  - Moving from standalone into cluster design supported by Redis

    - We should define SLA according to the number of records and performance

Metrics

Using external service in order to collect those metrics and monitoring the system (e.g Prometheus and Grafana)
The relevant metrics we'll collect from those services:

- Average number of requests for each service
    - Can point us if there is some problem in handle time
    - Also help us to track the scaling triggers
- Average handle time for each service
    - Can point us if there is some problem in handle time
    - Also help us to track the scaling triggers
- Average number of hash collision
    - Point us that the algorithm isn't good for our use case
- Total number of times we weren't able to save url due to no id
    - Point us that the algorithm isn't good for our use case
    - The way we are mix the url in order to get different id isn't good
- Total number of times of invalid Urls
    - Point us on problematic tests
    - Maybe our documentation isn't good

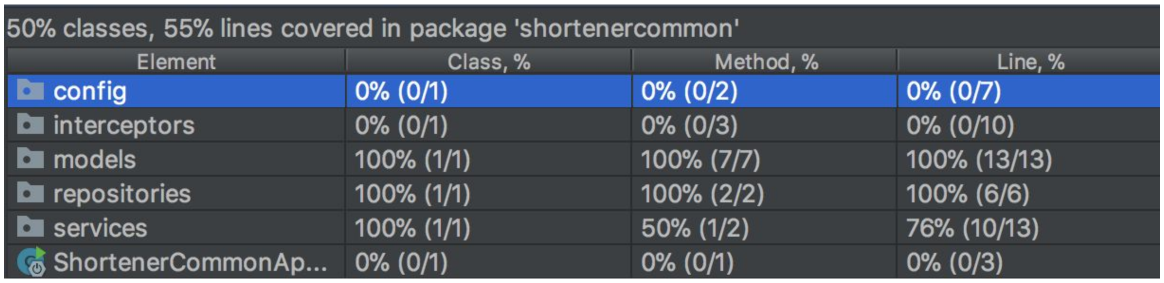
- Average number of unfound IDs
    - Can help us to detect DB problem
    - Some hacker attacks on our service

Production

Running a CI\CD pipeline(e.g. Jenkins, codeship) that will run the all the tests, the output will be runnable JAR that will be deploy to production (in case it's passed all) using rolling upgrade. Another option is to create at the end container with the JAR inside that will be upload to the cloud.

Test Coverage

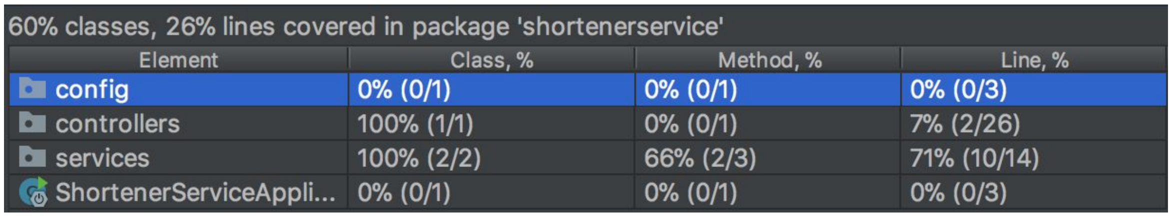
- Unit tests
  - Common



50% classes, 55% lines covered in package 'shortenercommon'

Element	Class, %	Method, %	Line, %
config	0% (0/1)	0% (0/2)	0% (0/7)
interceptors	0% (0/1)	0% (0/3)	0% (0/10)
models	100% (1/1)	100% (7/7)	100% (13/13)
repositories	100% (1/1)	100% (2/2)	100% (6/6)
services	100% (1/1)	50% (1/2)	76% (10/13)
ShortenerCommonAp...	0% (0/1)	0% (0/1)	0% (0/3)

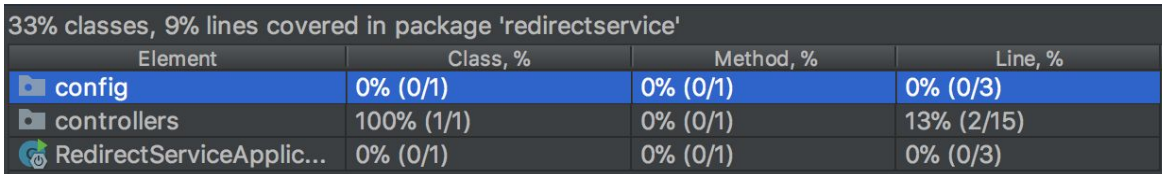
  - Shortener service



60% classes, 26% lines covered in package 'shortenerservice'

Element	Class, %	Method, %	Line, %
config	0% (0/1)	0% (0/1)	0% (0/3)
controllers	100% (1/1)	0% (0/1)	7% (2/26)
services	100% (2/2)	66% (2/3)	71% (10/14)
ShortenerServiceAppli...	0% (0/1)	0% (0/1)	0% (0/3)

  - Redirect service



33% classes, 9% lines covered in package 'redirectservice'

Element	Class, %	Method, %	Line, %
config	0% (0/1)	0% (0/1)	0% (0/3)
controllers	100% (1/1)	0% (0/1)	13% (2/15)
RedirectServiceApplic...	0% (0/1)	0% (0/1)	0% (0/3)

  - Note: I did implement the tests on the controllers but currently the have problem to run, therefore they should run under in Integration. Currently they are marked out

- Integration tests - Need to implement the following tests
  - Redirect service
    - Valid redirect - id exist in db
    - Not valid redirect - if not exist in db
    - Request with no db connection
  - Shortener service
    - New url, return new id
    - Url that already exist return the same id
    - Request with no db connection
    - Request that will generate existing key, but different URL - Hash collision handle

- Performance test
  Using Jmeter to run those tests
    - Redirect service
        - How many request we can handle in 1 minute
        - 10,000 requests for redirect in to different websites
        - Scale up triggers
        - Scale down triggers

    - Shortener service
        - How many request we can handle in 1 minute
        - 1,000 requests to create short url
        - Scale up triggers
        - Scale down triggers

Open issues

- Create runnable jar - currently run on IDE (Intellij) - it seems like local environment maven problem
- Controllers unit tests are marked as a comment since they are making a problem to run as part of the solution
- Implement integration tests
- Perform performance tests