Short URL Service

Design and Implementation details

ShortURL service is a Restful service for managing short URLs and redirecting clients from a long URL. This is written in GoLang and uses Redis database.

Reasons for using Redis database include:

- Redis is an open-source in-memory database (that can also be persisted). It is well suited for low latency requirements such as analytics operations and is known to offer good performance at scale for distributed applications. However, likelyhood of a data loss is higher with Redis than ,say, MySQL
- Redis also acts a message broker. Clients can publish and subscribe events
- Redis supports key expiry which is well suited for maintaining real-time and historical stats.

The service has three main operations, which are implemented as separate handlers in the code.

PostURL: This handler takes in a long URL. A global counter is maintained in the redis database, and each long URL is mapped to an integer number. This integer is converted to a short string of length 6 using the method described in (http://hashids.org/). Short strings encoded with this method are not easily discoverable. Incrementing any string has low probability of finding another valid string.

VisitURL: This handler takes in a short string and decodes it to an integer number. A long URL is retrieved from redis by looking up this integer number. The client is then redirected to the actual URL. This is a HTTP Get method

GetCount: This handler takes in a short URL and reports the number of times this URL was visited in the last 24 hours, 1 week and all time. This is a HTTP Get method. Redis supports key expiry. 3 counters are maintained for each URL which expire in 24 hours, 1 week and at infinity. A goroutine “ExpireWatcher” subscribes for expiration events and re-initializes the counters upon their expiry.

Building the system:

Requirements:

- Go (This service is verified on go1.9.1 linux/amd64)
  - External Packages needed: (go get github.com/<>/<>)
    - github.com/go-redis/redis
    - github.com/gorilla/mux
    - github.com/speps/go-hashids
  - GOPATH must include the current directory. For example, if the current working directory is ~/go, the codes must be placed in ~/go/src/shortenURL. GOPATH should include ~/go. GOPATH should also include the external packages directory which by default is ~/go.
  - GOROOT must point to installation directory of Go if it is different from the default location

- Redis-server (This service is verified on v4.0.2, 64 bit)
  - Config file: The server needs to be run with a redis.conf file (attached). It can be run as : redis-server <path-to-redis.conf>

- Rest Client

Ports 6379 and 8000 must be free to use.

Running the system:

Redis-server can be run as: redis-server <path_to_redis.conf>
The web server can be run as: go run <path_to_shortenURL_directory>/main.go

Testing the system:

A Rest client can be used to make the following requests.

- To obtain a short URL from a given long URL, a HTTP **Post** request can be made as:**http://localhost:8000/<long URL>**
- To access a web page using the short URL, a HTTP **Get** request can be made as:**http://localhost:8000/<short URL>**
- To obtain statistics for number of times a URL is visited within last 24 hours, 1 week, and all time, a HTTP **Get** request can be made as: **http://localhost:8000/<short URL>/count**
- The web server or the redis server can be restarted to test for persistence. The redis server must be restarted with the path to **redis.conf**.
- Developer tools in the Rest client (Firefox or Chrome) can be observed to verify for the time taken for redirection. This can be found in ‘Network Tab’ in Firefox. The entry will have a **GET** method with status code 302.