Canary: design document

Arvid Gotthard

High level description

The program I intend to implement is called **Canary** and is a *distributed key-value cache*. Essentially it is a associative array (hash table) that is split among separate data-shards (a server) in a cluster. Each of the shards will maintain an in-memory LRU cache with element expiration. A client application will execute a `get`, or `put` to one of the shards. In order to know which shard the client should fetch from

Requirements

Functional Requirements

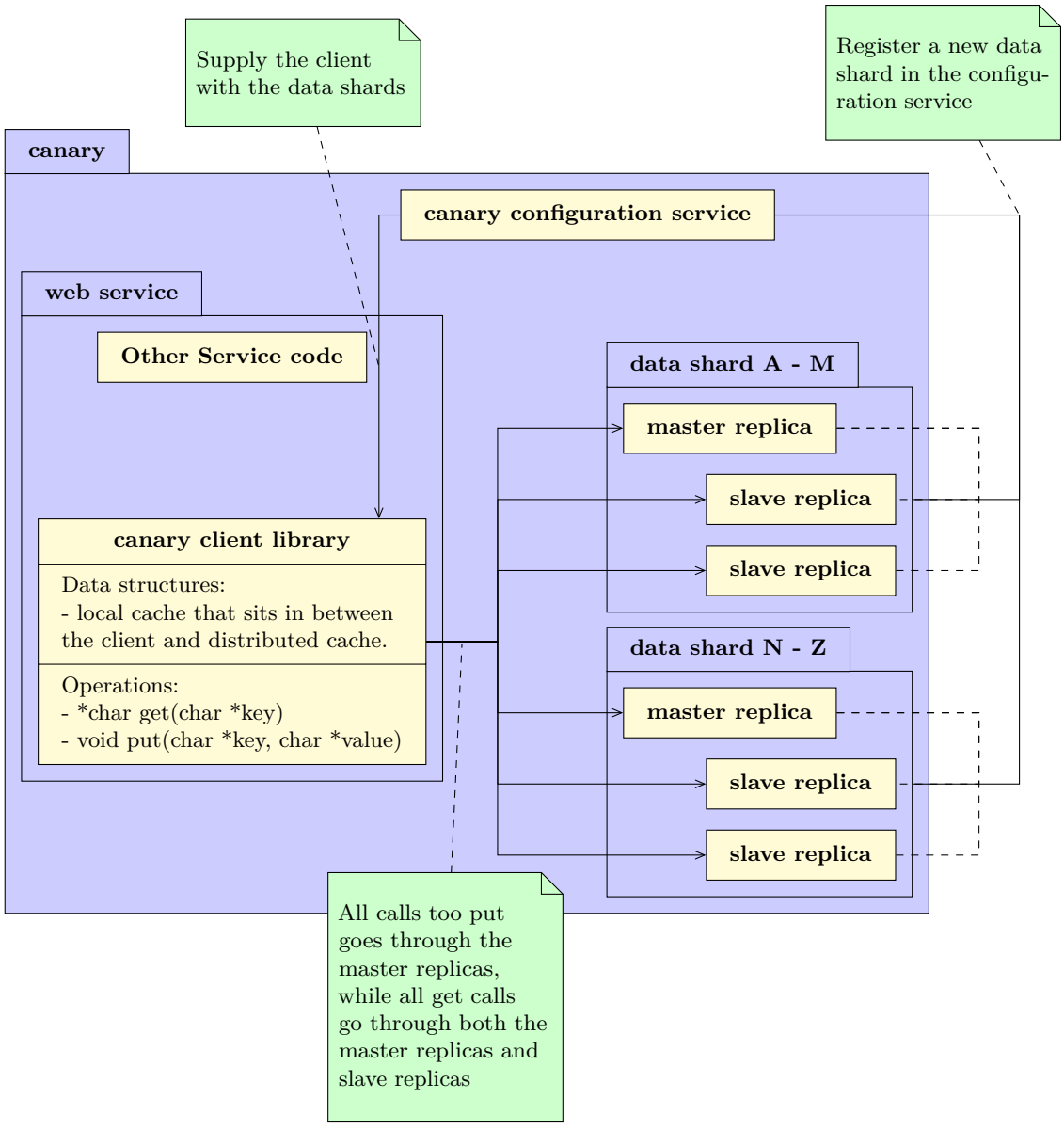
- `get(key)` - this operation will try to fetch from the cache. Will return a pointer to a `value` if there is a cache hit, and `null` in case of a cache miss.
- `put(key, value)` - this operation will insert a value into the cache.

Non-functional Requirements

- The main purpose of the cache is to be fast, as it is meant to sit in-between a web service and database
- We want it to be scaleable, so in the case we notice that we run out capacity we can add more data shards and the hashes can be redistributed among the shards.
- We want the system to have high availability, so in the case that a shard fails we have a replica that can be rotated in.

Design

Diagram



Design description

The system consist of 3 components, the client library, the configuration service and the data shards.

Client library

Client library is meant to be embedded in some sort of service or application. It in turn talks to the configuration service which will supply it with a ordered (by responsible hashes) list of data shards. The client can then easily search through the list using binary search. Once the client knows which data-shard to talk to it connects to them over TCP to do a `get` or `put` operation. The client can maintain a local cache to avoid making repeated calls to the distributed cache.

Configuration service

This service has the responsibility to register and assign which data shards are responsible for which range of hashes (will make use of consistent hashing). It will communicate to the client the list of data shards. It will also have responsibility to promote slave replicas to master replicas in case of failure of the master replica.

Data shards

Data shards maintain a local LRU cache in which the values are stored. If the cache capacity is exceeded it will expel the least recently accessed element. Each master replica will also run a thread to clean up expired cache entries. In the case a value in the master replica is updated or removed it the changes will be replicated to the slave replicas. Each replica in a data-shard will continuously send a heartbeat to the configuration service to signal that it is up.