Transparent distributed caching over REST

Jan Paul Posma, Roan Kattouw and Roy Triesscheijn

Introduction

We've written a distributed cache with the following properties:

- Cache nodes communicate via RMI to find the cache that has the information.
- Values can be stored and requested using REST over HTTP

Why caching?

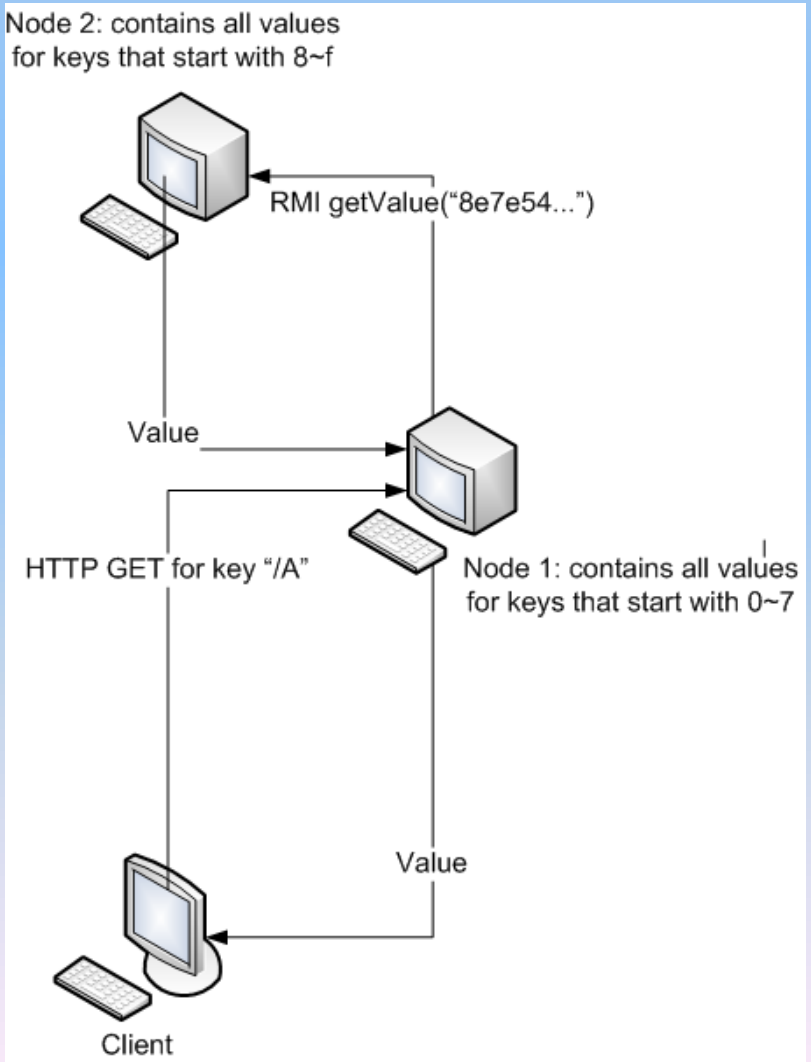
- For Websites and web application it is important that pages are served quickly.
- Information is stored in a database (all knowing but slow)
- Look up from a cache is faster. Data is stored in memory in an efficient data structure (HashMap in our case)

Why distributed?

- We can create a cache spanning multiple servers
- Or multiple caches on one computer
- We can easily add more computers to increase the memory size of our cache

Design

- Each node runs a basic HTTP server and is connected to other nodes using RMI
- Client issues a HTTP GET request with a key variable to a random node (Node 1 in this case)
- Node 1 hashes the key and finds out what node could have this information (we actually use a slightly different algorithm as pictured, using modulo)
- Node 1 requests information from Node 2 using RMI
- Node 1 sends “value” for given key back to the client
- Roughly the same for POST-ing of values



HTTP GET requests

- HTTP GET request format

When getting the value of a cache key the following HTTP request-response dialog takes place: (client, server)

GET /mykey HTTP/1.0

HTTP/1.0 200 OK
Date: Mon, 11 Apr, 2011 16:56:04 CEST
Content-Type: text/plain; charset=utf-8
Content-Length: 12
Connection: close

Hello World!

HTTP POST requests

- HTTP POST request format

When setting the value of a cache key the following HTTP request-response dialog takes place: (client, server)

POST /mykey HTTP/1.0
Content-Length: 12

Hello World!

HTTP/1.0 200 OK
Date: Mon, 11 Apr, 2011 16:57:58 CEST
Content-Type: text/plain; charset=utf-8
Content-Length: 0
Connection: close

Local cache

- All caches implement the interface ICache which require the classes to implement get, set and clear
- HashMap stores key-value pairs of strings
- LinkedList to throw away values when the cache is full

Remote cache

- We use ICache as a remote interface
- RemoteCache is the transparent handler for all caching requests, keeps a list of know servers and a local cache
- A HashBalancer is used to determine where to store and look for key/value pairs
- An RMI call is made when data is not stored at the local cache.

Real word examples

There are many real world examples:

- Memcached
- QuickCached
- SwarmCache