Key-Value Stores

Sometimes rDBs are too cumbersome ~~is~~ ~~[illegible]~~ when weighed against the services they offer. Also, they impose a tabular struc. on data viz not reqd. everytime. In those cases, we use non-rDbs for more flexibility and ease

Key-Value Store → i) A popular non-rDb (i.e. noSQL)
ii) Data stored as k-v pairs.
iii) Allows mapping of keys (~~strings~~ usually strings) to ~~values~~ arbitrary values.

eg1:

Key	Value
foo	-48·8
bleh	ApplePie
whut	[x, y, z]

eg 2: hashTables

k-v store
- → flexible → not rigorous as no imposed struc.
- → simple → nothin simpler to store data compared to a k-v map.
- → fast

Use:

1) Caching : Values → Responses to network reqs
Keys → Hash, IP address, username

Date:
M T W T F S

2) Dynamic config : Basically just storing a parameter that diff. parts of your sys. rely on in a separate place so all these parts can access it

eg:- usin config^n when you wanna switch what code you wanna run dependin on what envmt. you're in → (development, prod^n, testing)

Here, this param. is stored in a k-v store (usually an obj. in .json or .yaml file) ~~and~~ as workin with diff. values of this same param (key) is a lot easier

NOTE: Advantage of usin a k-v store → ∵ values are accessed directly thru keys, no search thru dB or any other fancy lookup reqd.
∴ Latency ↓, Throughput ↑ of our sys.

eg:- DynamoDB, Redis, Etcd, Zookeeper

NOTE : ~~Storage~~ 2 types of k-v stores based on where they write their data to :

i) Write data to disc → a) Slow to get back data from disc
b) Data persists even after k-v store server itself crashes

ii) Write data to mem → a) Faster to retrieve data frm. mem
b) Data lost if k-v store server crashes

Use :- Cachin → As honestly, it doesn't affect the sys much if caches crash but we do want readin frm cache = fast

~~[illegible]~~

∴ When a cachin involvin k-v store goes down, we pretty much don't lose anythin except some cache hits viz not that imp. a loss.

---

eg' of k-v stores:

1) Etcd
- → strong consistency
- → high availability
- → Use: Implement leader elecⁿ on a sys.
- → writes data to disc

2) Redis
- → writes data to mem. a.k.a 'in mem. k-v store
- → V. lil persistent storage
- → Uses: Best-effort, fast caching
  Implementaⁿ of 'rate limiting'

3) Zookeeper
- → writes data to disc
- → strong consistency
- → high availability
- → Use: Store imp configⁿ
  Implement leader elecⁿ

server.js — key_value_stores
server.js
database.js
server.js > app.get('/nocache/index.html') callback
Purchase | AlgoExpert
algoexpert.io/purchase
AlgoExpert
Ace the Coding Interviews
Contact Us | FAQ | Reviews | Become An Affiliate | Legal Stuff | Privacy Policy
Ln 7, Col 47
Spaces: 2
UTF-8
LF
JavaScript
Prettier
07:11
algoexpert.io

```javascript
const database = require('./database');
const express = require('express');
const redis = require('redis').createClient();

const app = express();

app.get('/nocache/index.html', (req, res) => {
  database.get('index.html', page => {
    res.send(page);
  });
});

app.get('/withcache/index.html', (req, res) => {
  redis.get('index.html', (err, redisRes) => {
    if (redisRes) {
      res.send(redisRes);
      return;
    }

    database.get('index.html', page => {
      redis.set('index.html', page, 'EX', 10);
      res.send(page);
    });
  });
});

app.listen(3001, function() {
  console.log('Listening on port 3001!');
});
```

The Ultimate Platform.

Organized. Guided. All-encompassing. That's AlgoExpert.

You've purchased all available products.

Copyright © 2019 AlgoExpert LLC. All rights reserved.