APRIL 2022

WK	M	T	W	T	F	S	S
14					1	2	3
15	4	5	6	7	8	9	10
16	11	12	13	14	15	16	17
17	18	19	20	21	22	23	24
18	25	26	27	28	29	30	

(Module 4)

2022 FRIDAY 01
091-274 WK 14
APRIL 01.04.2022

9 Key value stores

10 (DHTs) distributed hash tables.
value could be anything.
11 are used in storing user sessions.

12 Traditional databases dont scale well.
~~if~~ for distributed nodes

Requirements — functional

(1) clients should be able to choose between availability and consistency.
(2) ability to always write > availability.

non-functional

(1) Scale ↑ high.
(2) Availability ↑
(3) Fault tolerance → operate despite failures in some components.

02 | 2022 SATURDAY
092-273 | WK 14
02.04.2022 | APRIL

APRIL 2022

WK	M	T	W	T	F	S	S
14					1	2	3
15	4	5	6	7	8	9	10
16	11	12	13	14	15	16	17
17	18	19	20	21	22	23	24
18	25	26	27	28	29	30	

Assumptions

1. authentication / authorization already done
2. data centers are trusted.
3. https → user requests and responses

API

get(key)

put(key, value) (can also store version some metadata of the object)

Adding scalability

~~additional~~ more storage as time progresses

nodes, (+, -) → moving of data around nodes. → consistent hashing

Adding availability → replication

03 SUNDAY

our primary goal is to design a system that can always write → hence no primary secondary replication. peer to peer → high write

async

Data versioning

in event of network partition, component(s) failure, inconsistent data can be formed.

resolutions:- ① Time isn't a reliable in ds).

② Vector clocks → (node, version).

every object has a version (A, 1)

this is written by node A, V→1

in case of conflicts → clients resolve it.

API design change of put

put(key, context, value);