Dynamo

Dist Sys

=> early key-value storage

Theory Academic → Practice

'00 → '10

web services

Goals

System Arch

(most)

Goals:

- → Scale ← shopping carts
- → Efficiency/Latency (tail)
- → Availability (over all other things)
- → Simple API : get/put keys/value

hard

=> rise of key/value stores

~"NoSQL movement" ↔ "SQL"

two religions

Background :

Availability : → ["eventual consistency"]

"always writeable" → make writes highly available

⇒ allow writes

conflict resolution on reads

⇒ how to resolve conflicts?

→ system ← e.g. last writer wins (overly general)

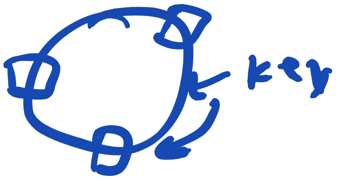
→ app can ⇒ & be smarter

System Architecture :

API : get/put ⇒ shopping cart ⇒ is this interesting? (adv.)

Partitioning

⇒ consistent hashing (but not full thing a la Chord paper)



(use virtual nodes)
+ ~~Rate~~

Replication : N hosts

1 of N: coordinator

put → make sure key properly replicated

"preference list" : nodes resp. for key k

N-way replication,

pref list > N : why?

→ N distinct phys nodes

→ may be some down

"Eventual" Consistency : replicas not always in sync.

put new

c_1: get → new

c_1: get → old

support for "shopping cart"

→ never lose "add to cart"

adds/deletes ⇒ puts (writes)

how to track versions/conflicts:

version vectors

⇒ easy conflict detection

↳ standard problem: can be large

limit size to 10

Executing get/put:

1) Load balancing

→ generic load balancer (might need extra hop)

→ partition-aware client library

↳ why better?

2) coordinator:

first N healthy nodes (w/ failures → go further down pref list)

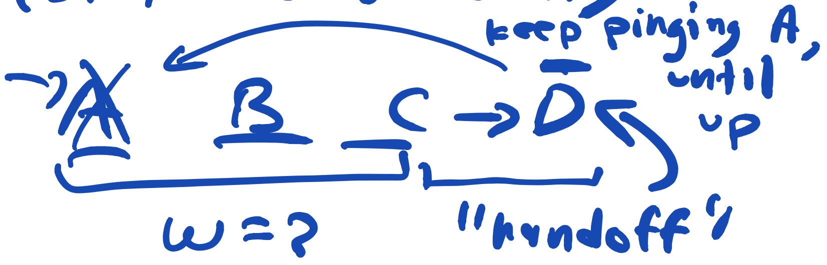
Quorum-based: R, W

if R+W > N, intersection

but: availability

instead "sloppy" quorum:

=> first N healthy nodes
(skip nodes down)



Replica Synchronization

=> Anti-entropy { Random pairwise sync }

=> Detect/fix differences:

[Merkle trees]

problem: classic "consistent hashing"

=> key ranges keep changing w/ node addition/deletion

(Break)

Ending the class :

	M	T	W	Th	F
			~~5th~~	Dynamo.	7
	[RamCloud] 10		12	(auth) BlockChain	14
mid prep	17 [Midterm #2] take home		19	20	21

BigTable

Ceph/Sage update

(final pres (+ if desired very short writeup)) + [celebration]

Dynamo: last tech piece

membership: join/leave

worried partitioned: solve by using "seed" nodes

failure detection: mostly to "local" failure detection

e.g.) coordinator → replicate write

$x_1, \boxed{\times} \ldots \rightarrow x_N$

historical: p2p

not p2p: smaller scale

Experience / Lessons:

Performance vs. Durability:

(trade-off) (latency)

e.g. in unix file system

disk / OS cache: write() → mem

Dynamo:

write buffer: batch locally (perf ⇒ reduced tail latency by 5x)

⇒ can lose data

writes to N nodes:

→ memory

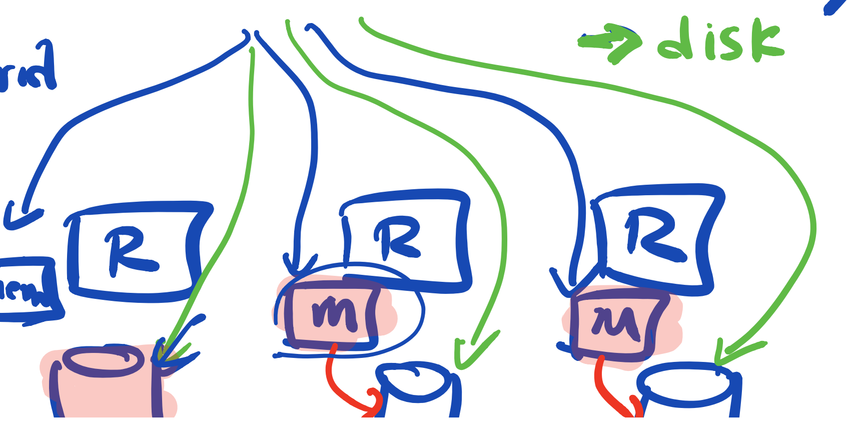
→ disk

Dynamo: hybrid

→ N node rep.

→ 1 durable mem

rest (w)



memorys
(eventually persisted
memory:
perf but
not as
"reliable"

disk:
other
way

→ but: make more reliable by
→ placement (across racks,
datacenters,
etc.)
(but bugs...)

Load Distribution

⇒ Partitioning:
Consistent Hashing
⇒ really simplified
(as compared to
(Chord)

why?
1) ⇒ [Chord: variable size
ranges]
→ hard to sync.

Background Tasks

⇒ scheduling
of bg
tasks
(details:
not shown
here)

instead: fixed-size partitions

2) knowledge of key-range responsibility

→ Chord: distributed

→ Dynamo: all nodes have this mapping info

Conclude:

→ key/value (NoSQL)