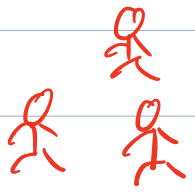
Agenda:

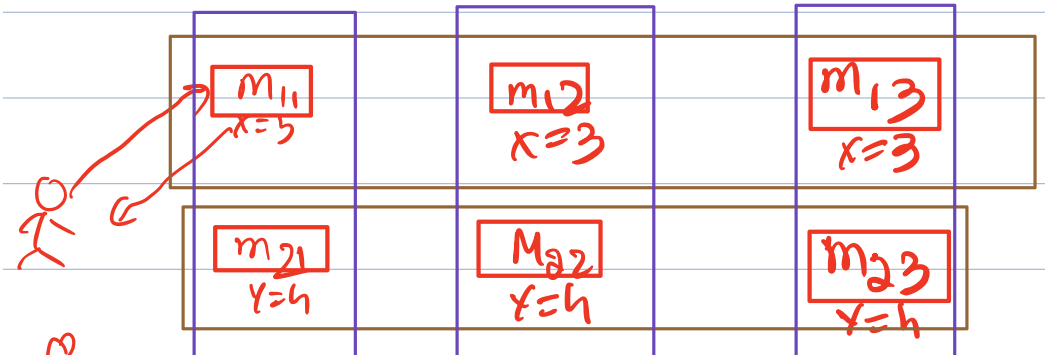
- Intro to sharding (data partitioning)
- consistent hashing

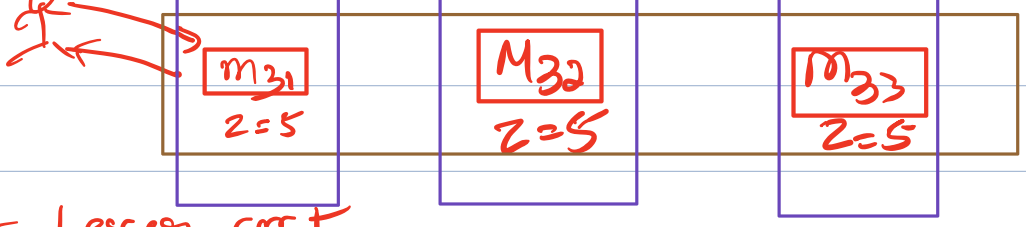
Riak - Inspired by DynamoDB
- Open source clone of Dynamo

	m_1	m_2	m_3
Clients			
	$x=3$ $y=4$ $z=5$	$x=3$ $y=4$ $z=5$	$x=3$ $y=4$ $z=5$

Whats wrong with an approach where everyone stores all of the data?

- What if you have more data than can reasonably fit on one machine?
- If everyone stores all data, consistency is more expensive to maintain.





- Lesser cost
- Split data across machines
- Save effort of keeping consistent
- Distribute load

What did you lose by changing design?
- No fault tolerance.

Sharding & Replication are orthogonal to each other

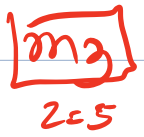
Also called data partitioning.

Why would you do this?
- capacity increases.
- throughput increases

Sharding

How do we decide how to split up our data among shards?

What makes a sharding strategy good?



Randomly assign data to nodes?

✓ - evenly distribute data

☹ - hard to find what we want

Put it all in one place?

✓ - easy to find it

☹ - the worst at even distribution

Every partition stores a range of keys?

☹ - Distribution might be uneven

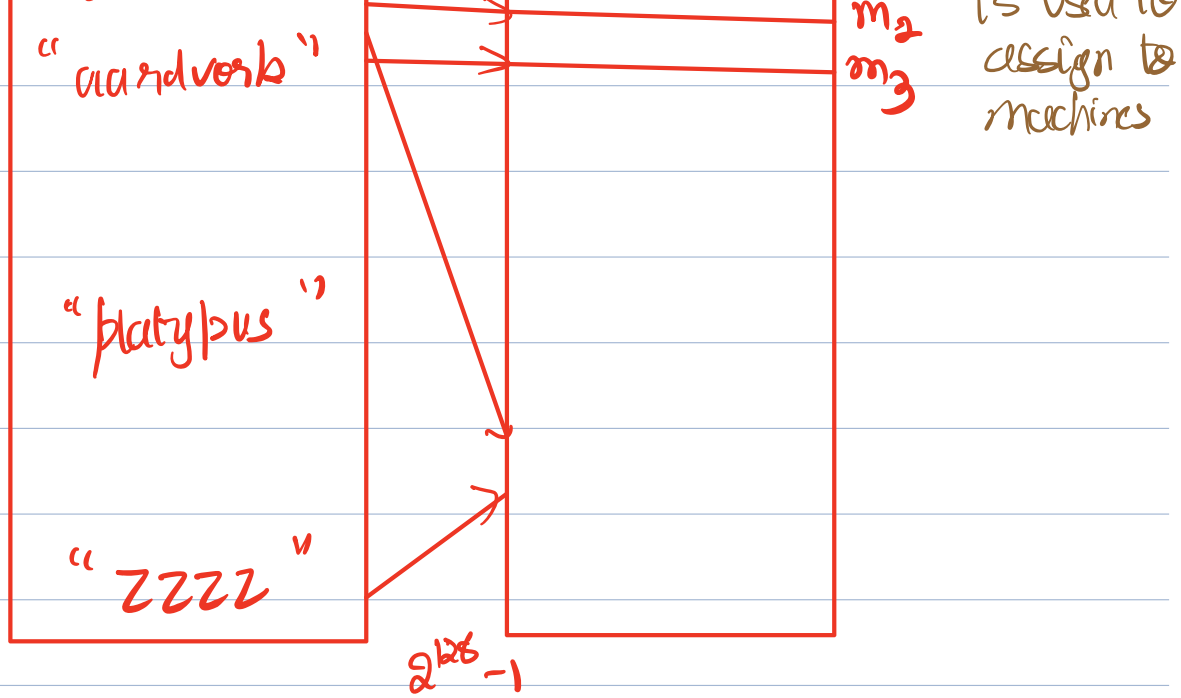
How do you take data that is not uniformly distributed & make it uniformly distributed?

Hashing!

Hashing

MD5: input → output

Assuming hash % 3



m_1 "aaaa"

m_2 "aardvark"
"apple"
"ZZZZ"

m_3 "awkward"
"platypus"

Partitioning by hash of key % N, where N = # of partitions.

What is the catch with the above approach?
- Assumes that N is fixed!
- If N changes, most of the data needs to be moved around!

Assume, MD5 ("aardvark") % 3 = 1
MD5 ("aardvark") % 4 = 2

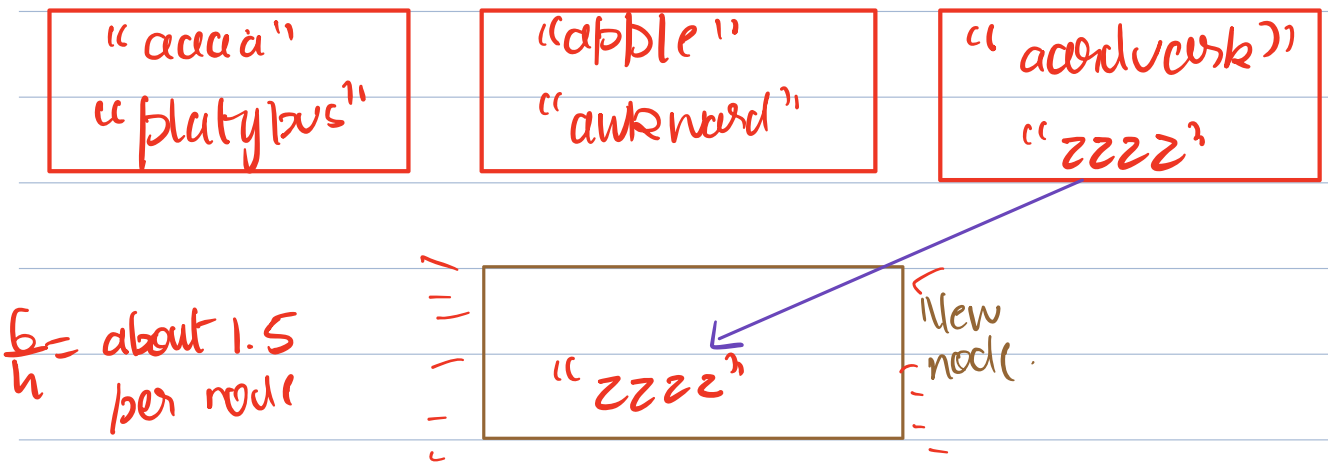
Even if there is a new machine, the key is moved to another old machine !!

∴ Too much data movement when N changes

what is the minimum acceptable movement?

If there are 6 keys split across 3 machines, on average, each machine should have 2 keys

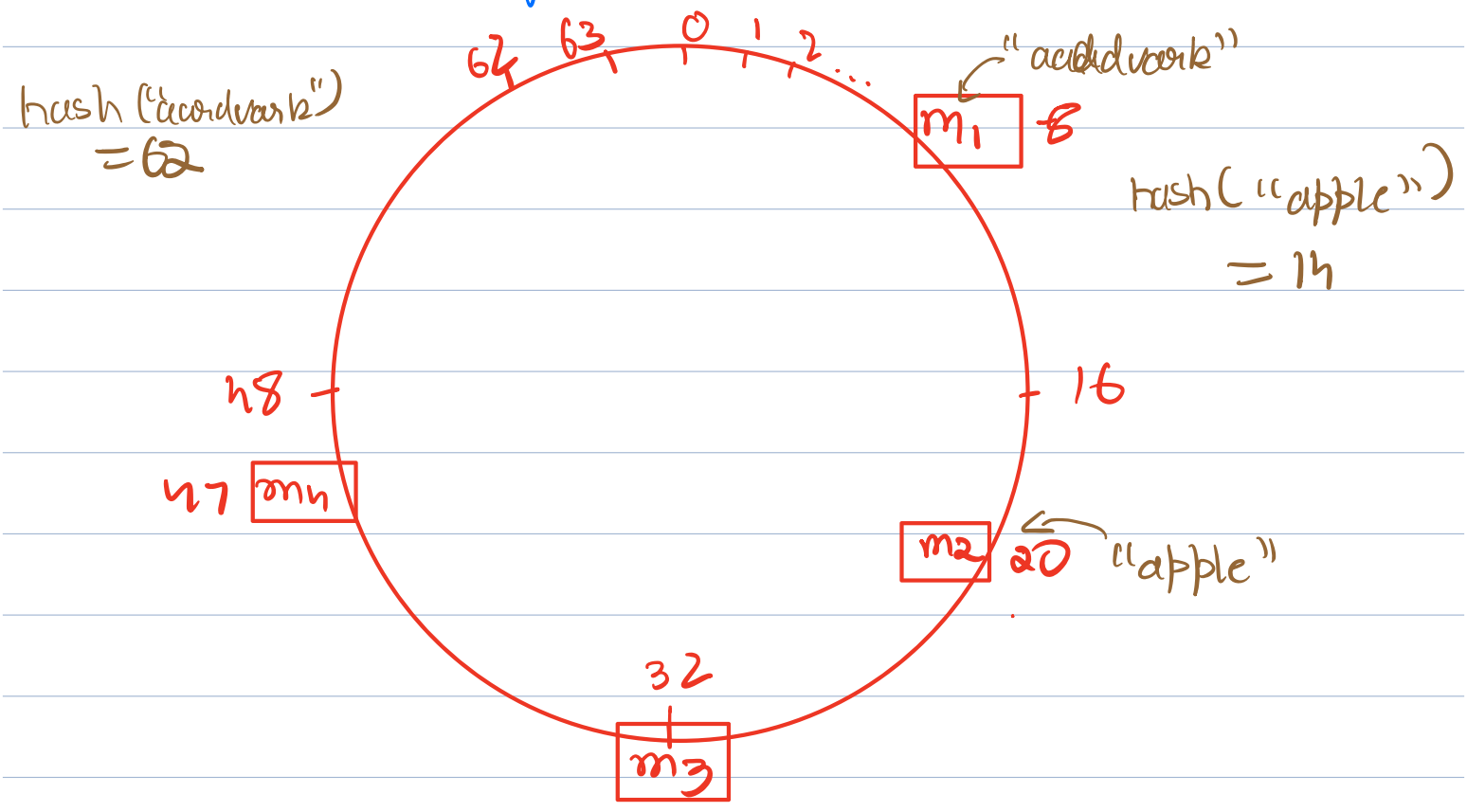
h machines ⇒ 1.5 keys



For K keys & N nodes, $\frac{K}{N}$ is the minimum movement possible to get an even split

"Consistent Hashing" gives us this minimum movement

Consistent Hashing



- Nodes are organized around a ring
- Assuming max no. of slots on the ring is 2^6-1 i.e the hash function used gives 2^6-1 values, each node exists on a particular slot
- How do you determine the slot for each node?
  - Maybe you can hash the IP address of the node

- Suppose you want to store the key "apple". After hashing, we get the value 14
- Since there is no node on 14, we will walk the ring clockwise & store it on the first node we see. Hence, "apple" is stored on

node M_2.

How does Dynamo use consistent hashing for replication?

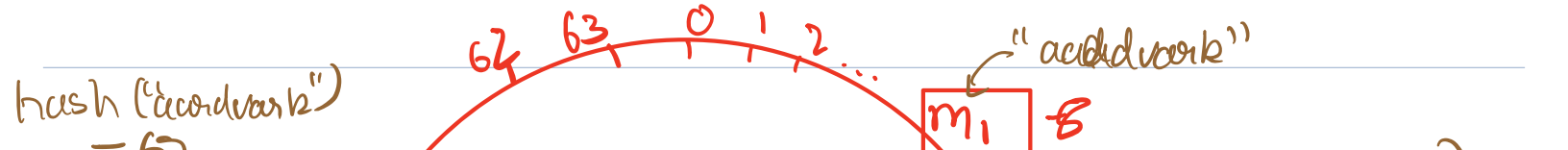
- Assuming the number of replicas is 3.
- The original hash value of the key is used to store the data in a node called the Home Node.
- Since the replicas is 3, we will walk the ring clockwise & store the key in the next 2 nodes. This will be m_3 & m_4

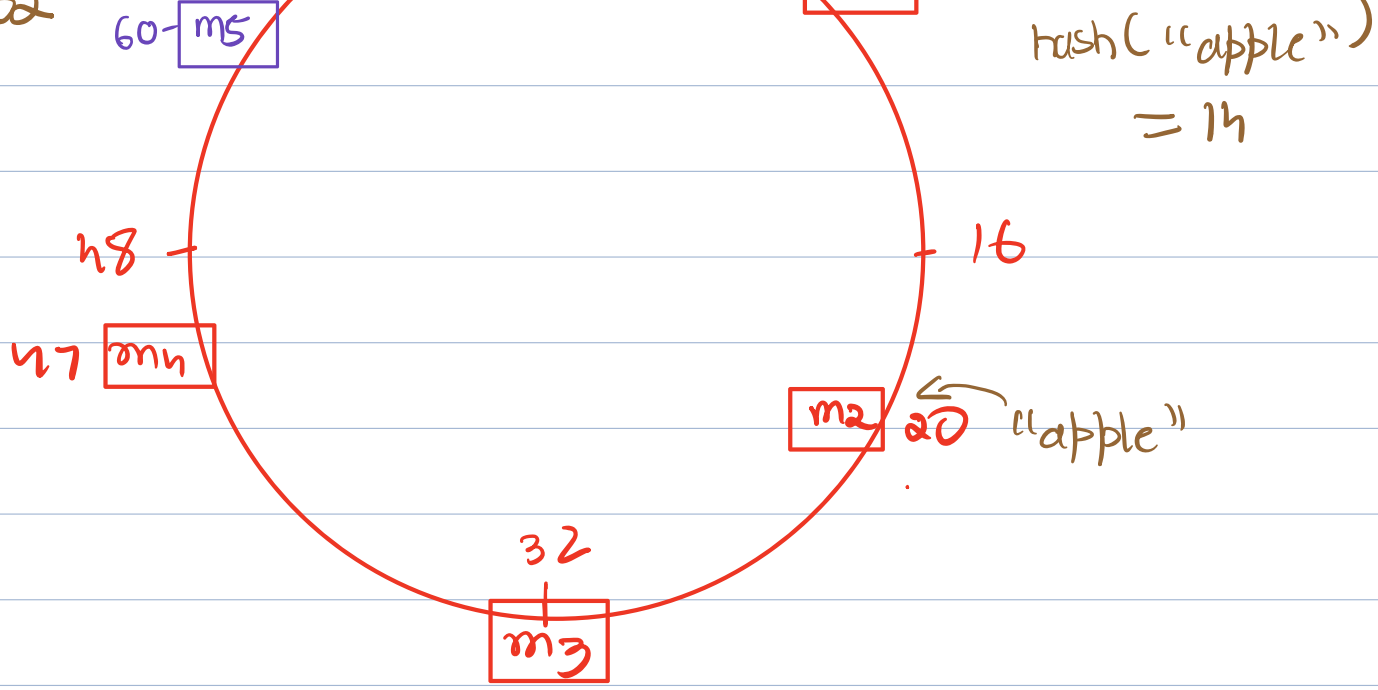
In dynamo, every node has a preference list i.e. list of nodes where the data should be replicated.

Generally the preference list is longer than the replication factor. Why?

Some nodes could be unavailable for some reason.

How does consistent hashing help reduce data movement?





New node M_5 at 60. Which keys do we have to move?

If hash("zzzz") = 50, before M_5, it would go to M_1

After M_5, M_1 is responsible for smaller range.

∴ Anything that would have hashed to M_1 initially when the value fell in the space for M_5 (48 - 60) would have to be moved to M_5

However, on average, $\frac{K}{N}$ keys will be moved

All other nodes don't care about this data movement !!

What happens if a node crashes?

Suppose M_2 crashes

M_2 was responsible for range (9-20).
Since M_2 is gone, M_3 needs to store more keys

However, all other nodes are unaffected!

Hopefully, M_3 already has the backup of these keys, and can act as temporary home node for these keys before M_2 replacement is up.

How/When can consistent hashing go wrong?

- Human needs to monitor system health & ensure not too many nodes crash.

- If couple of nodes crash, neighbor has to pickup lot of slack.

- In the case where nodes are not evenly spaced, the O/P space may not be evenly covered

Solution?

Hash a node to many spots)

Same node, but it is available on many spots. These nodes are called virtual nodes.

Advantages

- This makes it more likely that nodes are evenly distributed on the ring.

- Virtual node count can be proportional to the capacity of each node.

  M_1 has 10 TB space ⇒ no virtual nodes

  M_2 has 4 TB space ⇒ 16 virtual nodes

  Helps integrate heterogeneous machines

Disadvantages:

- If M_1 goes down, virtual nodes are down too. Lots of other nodes need to pickup slack from other parts on the ring.

In Dynamo, this is stated as an advantage as data movement happens between different machines.