→ Last two classes

① Serverless computing

② Ex Camera – video processing / encoding
Starling – query processing

→ Today's class

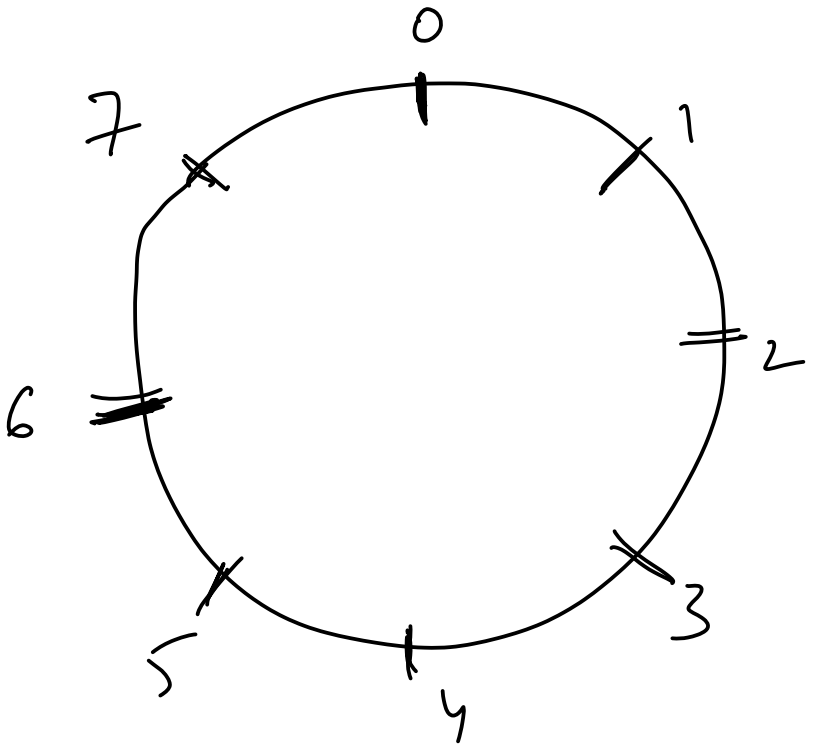
Consistent hashing / Distributed Hash Tables

① Classical distributed system technique utilized in a wide variety of large scale distributed systems

e.g. Dynamo (AWS)
↳ storage
Parameter server (ML)
↳ partitions the model parameters

partitions the data objects

Chord distributed hash table [2001]



$m = 3$

$2^3 = 8$

① Distributed because data is hashed or partitioned across multiple nodes or machines

② Each node is assigned an ID.

$$\text{NodeId} \in [0, 2^m - 1]$$

m bit string $\Rightarrow$ 2^m max. no. of nodes

* We hash the (IP address + port) to a m bit string

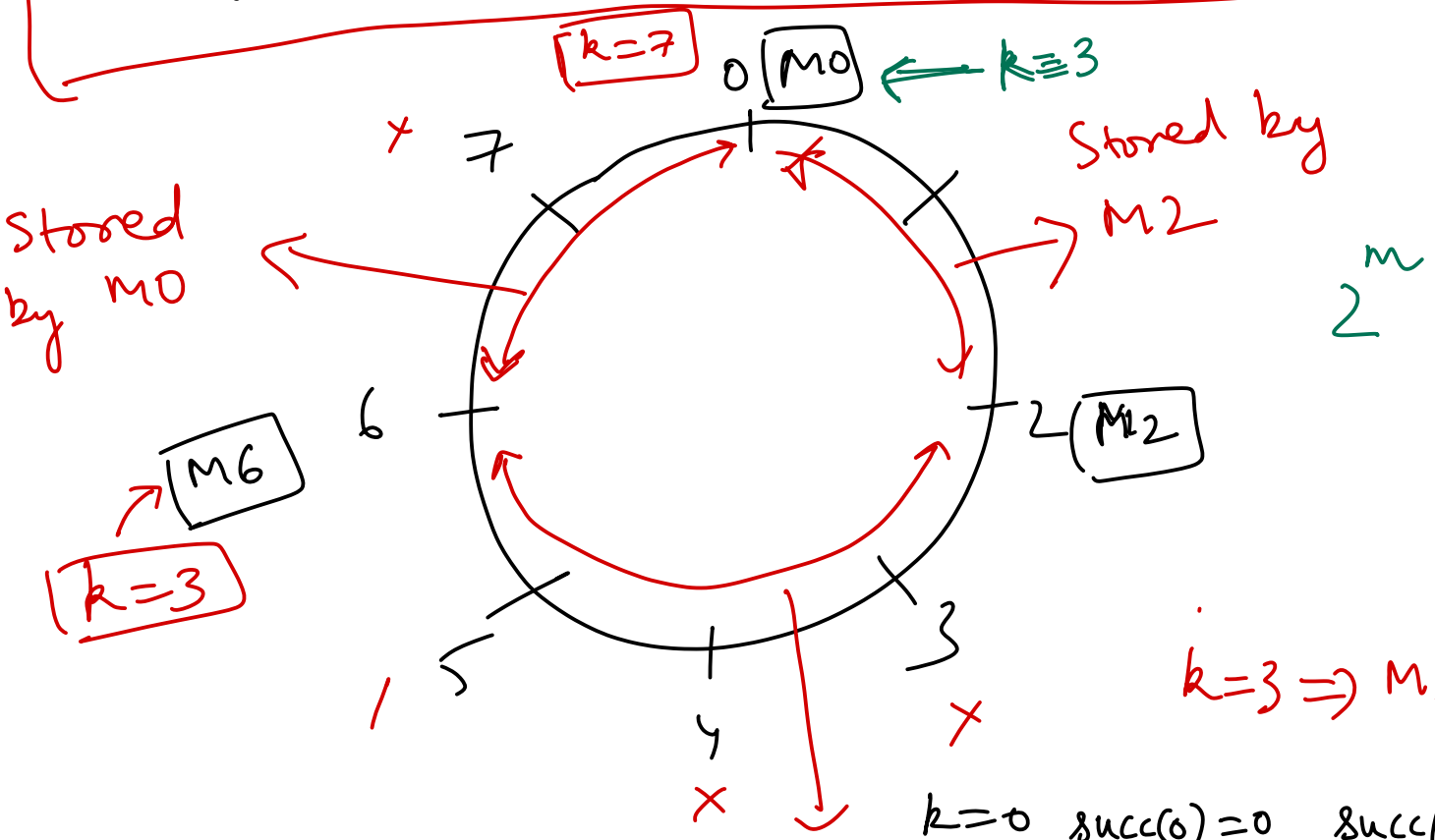
③ Each data item/object is also hashed to generate a key b/w 0 and $2^m - 1$

$$h(x) \Rightarrow \text{key} \in [0, 2^m - 1]$$

h → Hash Function, x → data item

$succ(k) \geq k$

Data item with key k goes to the node with the smallest ID $\geq k$



$k = 3 \Rightarrow M6$

$k = 7 \Rightarrow M0$

$k = 0 \Rightarrow M0$

$k=0$ $succ(0)=0$ $succ(6)=6$

$k=1$ $succ(1)=2$ $succ(7)=0$

2 $succ(2)=2$

3 $succ(3)=6$

4 ...

Q Any request for storing a new data object/item or retrieving an existing data object can come to any node present in the circular ring. How do we efficiently search for the right node on which data is stored or should be stored

Simple lookup/search $O(1)$ → Space complexity
$O(N)$ → Time complexity

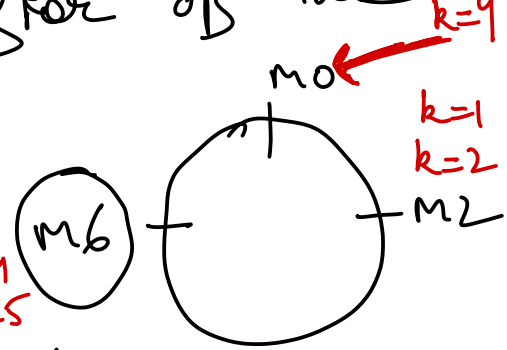
① Each node needs to store only the successor node details

② A query for key x is forwarded to the successors of nodes until it reaches the first node such that the node's identifier y is greater than x (modulo 2^M)

- High latency (O(N))
+ less space requirements (O(1))

Scalable lookup / search

(1) Each node n maintains a routing table called as the Finger table with at most O(m) distinct entries such that the x^{th} entry $(1 \le x \le m)$ is the node identifier of the node $succ(i + 2^{x-1})$



Finger table for i=0 (M0)

x	$i+2^{x-1}$	$succ(i+2^{x-1})$
1	1	succ(1) = 2
2	2	succ(2) = 2
3	4	succ(4) = 6

k=4

Finger table for i=2

x	$i+2^{x-1}$	$succ(i+2^{x-1})$
1	2+1=3	succ(3) = 6
2	2+2=4	succ(4) = 6
3	2+4=6	succ(6) = 6

k=4

i=6

x	$i+2^{x-1}$	$succ(i+2^{x-1})$
1	6+1=7	succ(7) = 0
2	6+2=0	succ(0) = 0
3	6+4=2	succ(2) = 2