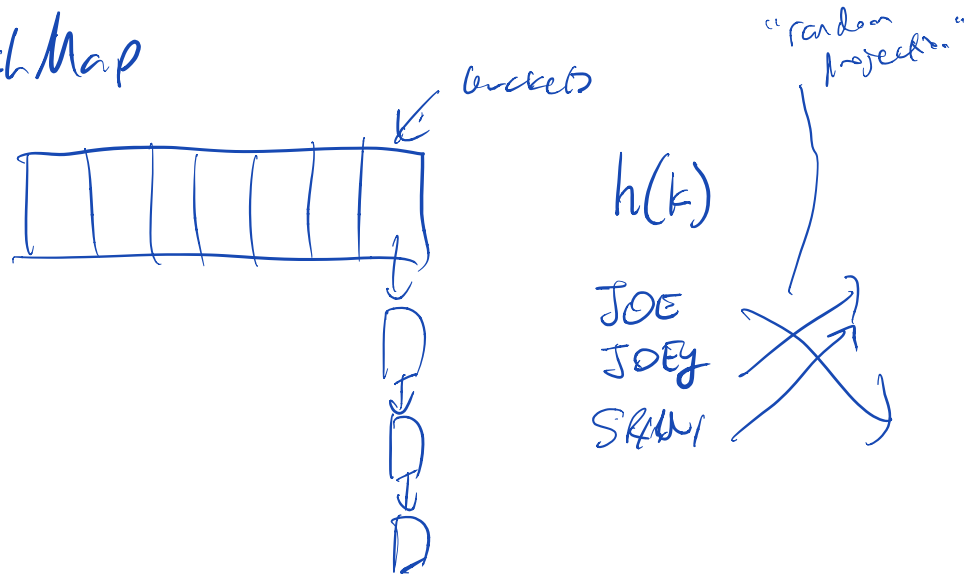


key value store
hash table

HashMap

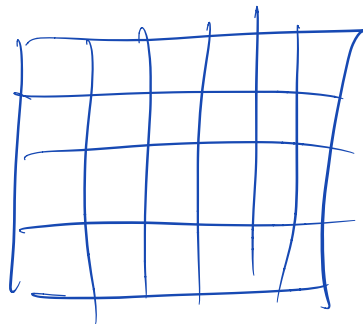


pigeonhole principle

M objects N bins "pigeonhole"

$$M > N$$

some bin
has > 1 thing



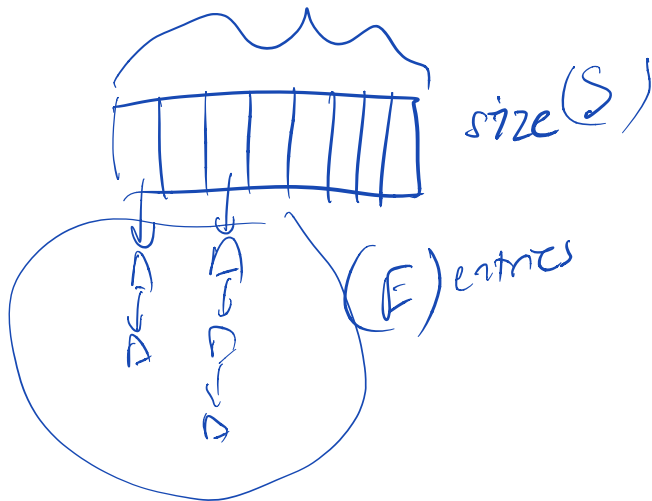
GOAL: $O(1)$ access time

$$h(k) = 0$$



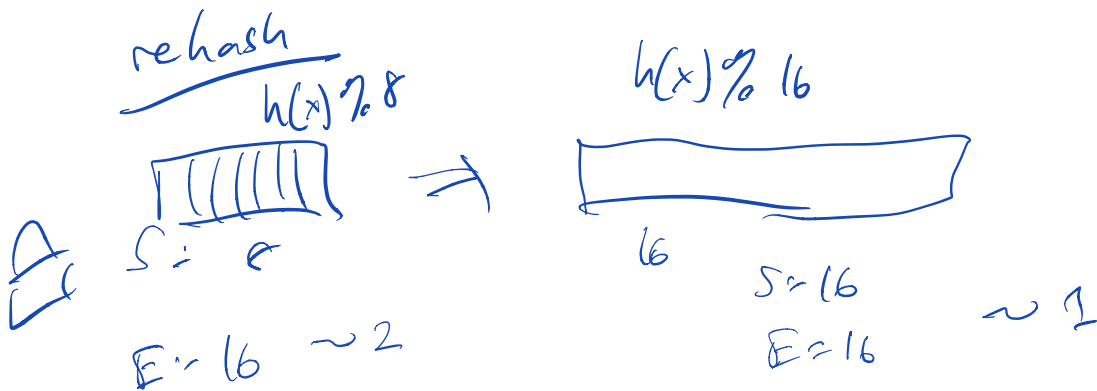


hash tables \rightarrow dynamic hash tables

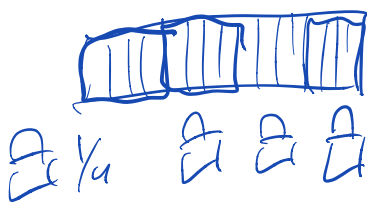


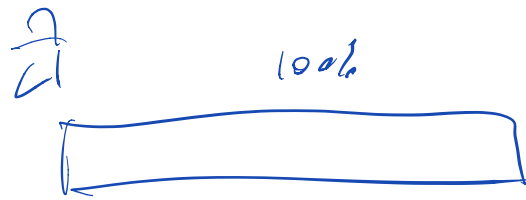
if
hash function
"good" =
effectively rand
distribution

length of traversal
 $E \left[\frac{E}{S} \right]$



Concurrent Hash Map

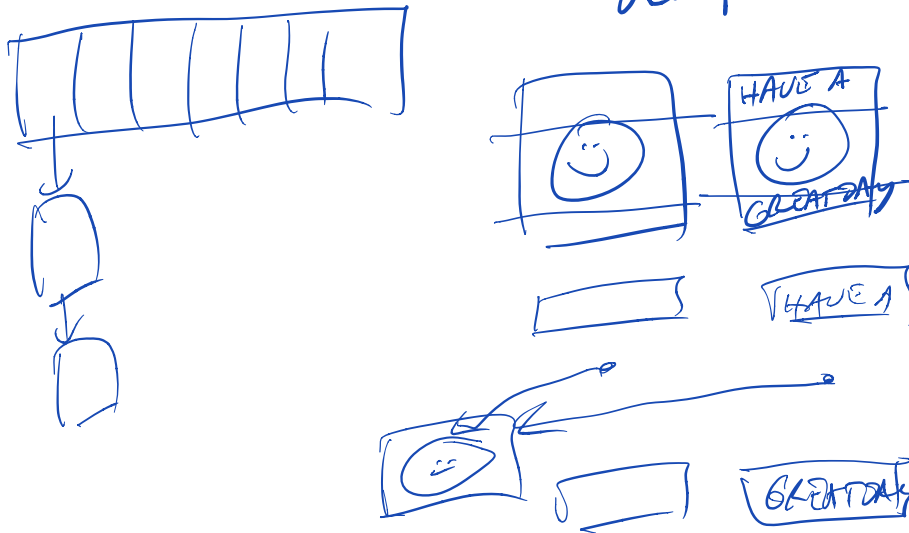




Java objects \rightarrow hashcode

$$\text{hash}(b_i) = \sum b_i = \bigoplus b_i$$

deduplication



hash \rightarrow "Fingerprints"

$$h(x) \neq h(y) \Rightarrow x \neq y$$

sum of ^{independent} identically distributed & rand vars.

\Rightarrow approx normal



MDS digest

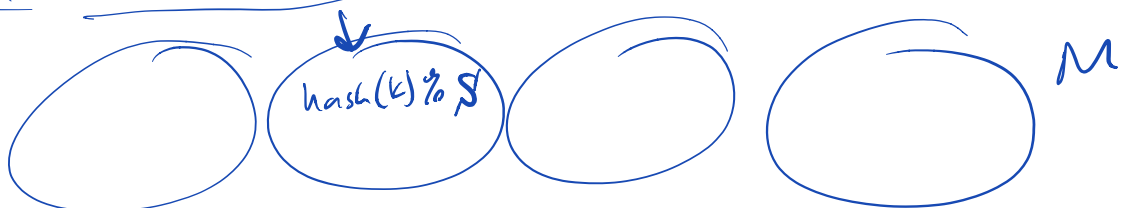
	xor	1
0	0	1
1	1	0

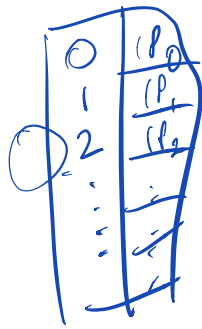
Distributed Hash Table (imp. l. key-value store)

① BIG

② FAULT TOLERANCE (SPoF)

hash(k) % Machines





GET(k) \rightarrow v
PUT(k, v)

PUT(k, v)
GET(k) \rightarrow v EMPTY
ERASE
...



%4



%5



Capacity



replica 

Now Network of Workstations

NO WAY

RAID
RISC

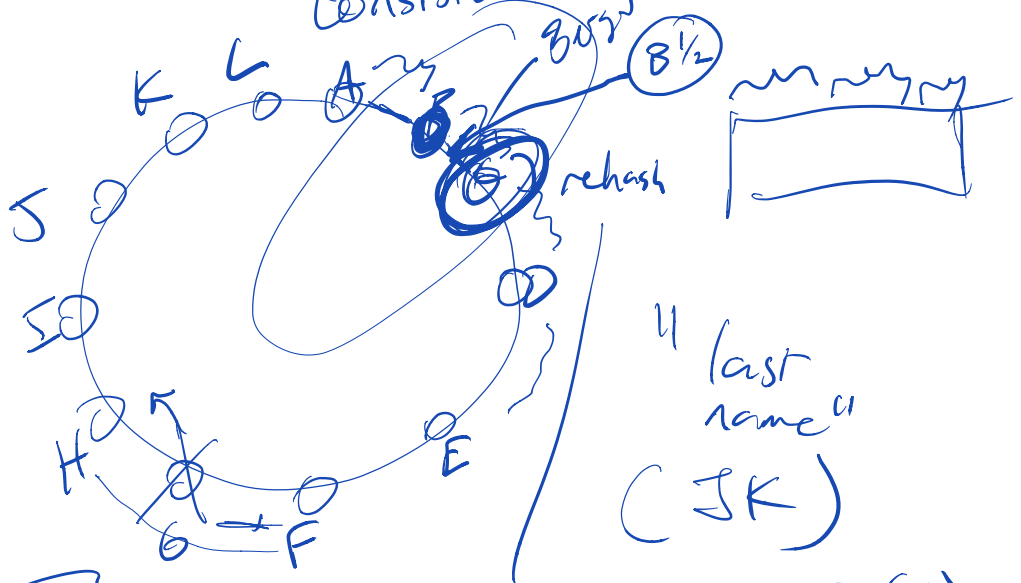
Dave
Patterson Berkeley

DHT

Chord

2001 MIT

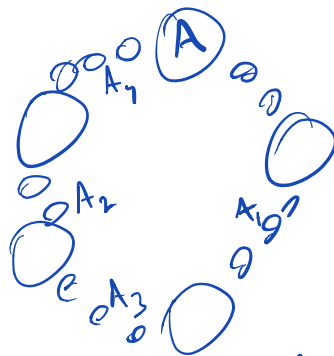
consistent hashing



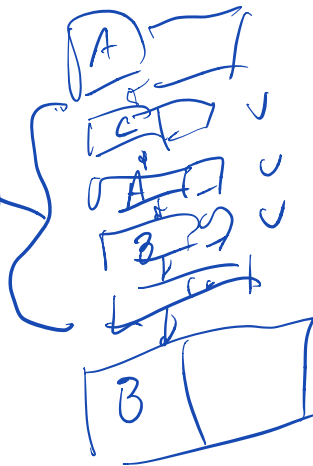
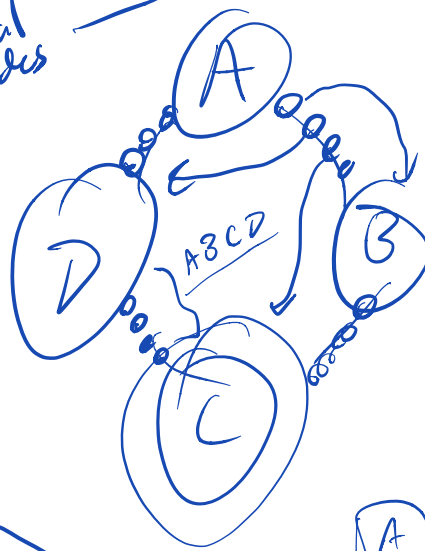
Chord
metadata

#rehash is not $O(N)$

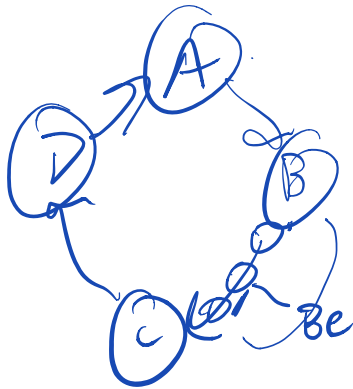
$\Rightarrow O(1)$
(machines)



Virtual nodes — spread load — esp. in case of failure



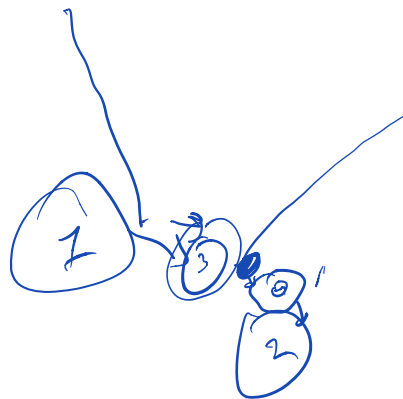
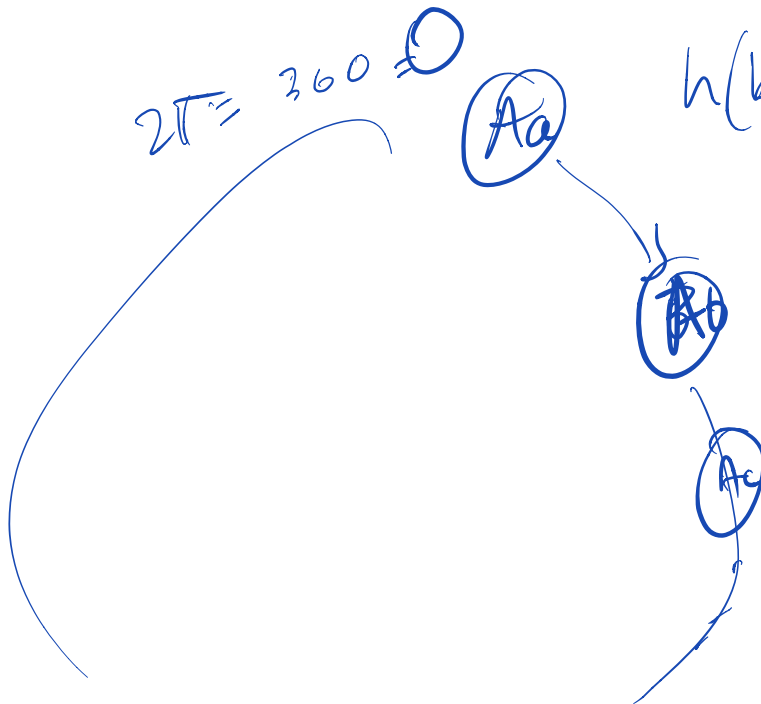
Berser



$2\pi = 360^\circ$

$h(k) \rightarrow \text{ }^\circ$

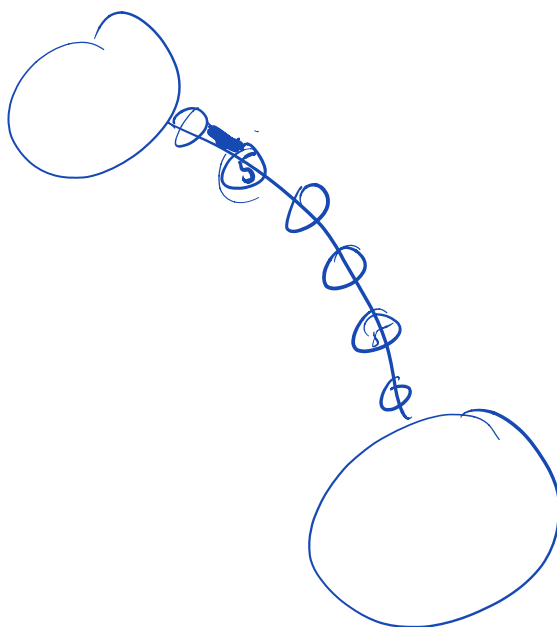
hash to
angle



5

3

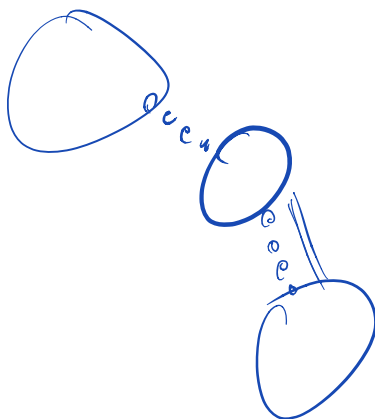
9



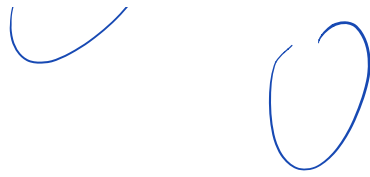
5 /

5

5



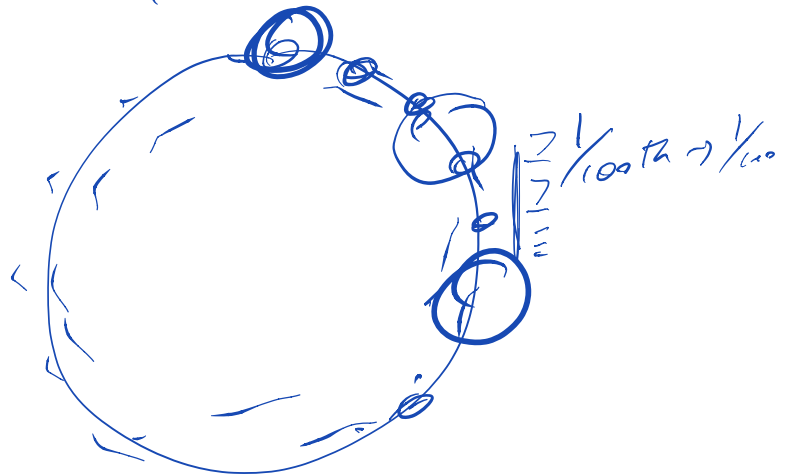
Adding



BIG
FT

— dynamic — grow — Scale

100 → 104



virtual nodes $\sim 1000^\#$ real nodes

Chord — DHTs
Scalable key-value

Bloom Filters ~ 1970



1 set membership queries

FIND(k) \rightarrow T
F

ADD(k)

$\{h_1 \quad h_{32} \quad h_{99}\}$

(100 people) h_w

[0 0 0 0 0 ... 1 0 0]

$O(1)$

bitset

OPTIMIZE

$|set|$


= 1 billion

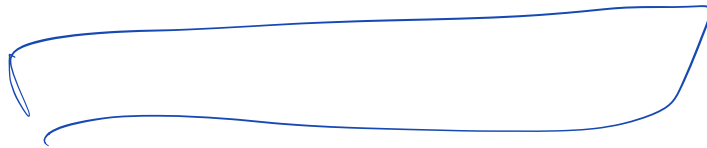
\Rightarrow 119 MB


Bloom filter — probabilistic data structure
trade size for accuracy

GET(k) ($k \notin S$) \rightarrow F

GET(k) ($k \in S$) \rightarrow MAYBE

 Bloom Filter
 TINY
 "cache"

 1BT

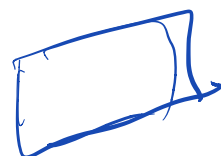
 1 BIT
 Bloom Filter

$h(x) = \text{PARITY}(x)$
 $\text{EVENBITS}(x) \rightarrow 1$
 else $\rightarrow 0$

$h(\text{me}) = \text{odd}(0)$
 $h(\text{Karl}) = \text{odd}(0)$
 $h(\text{Sam}) = \text{odd}(0)$

$h(\text{Len}) \rightarrow 1$

$h(\text{SamB}) ?$

$h(k) \% N$

 OR
 ↑
 PUT

GET

 Bloom Filter
 ~

10100