

## Consistent Hashing

Supongo que deseas tener un índice distribuido sobre  $n$  máquinas/servidores  $S_1, \dots, S_n$

Possible solución: Tener una función de hash:

$$h: D \rightarrow \{1, \dots, n\}$$

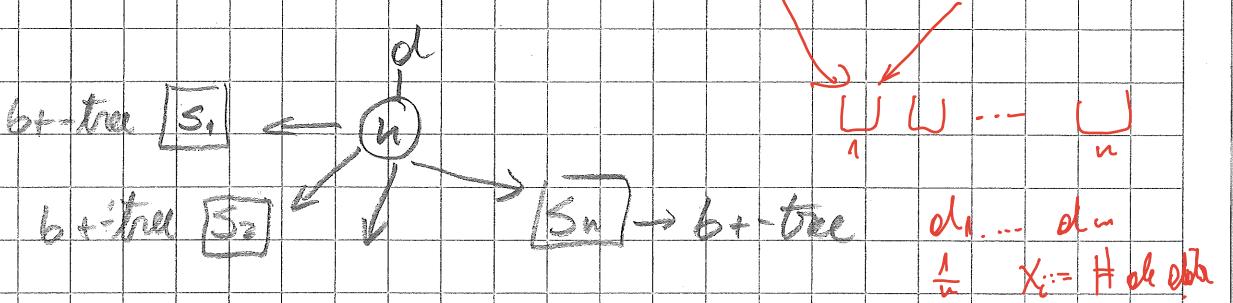
Tal que cada servidor  $S_i$  recibe el conjunto:

$$\{d \mid h(d) = i\}$$

Propósito: - ¿Cuál es la carga de cada máquina?

- ¿Cómo buscamos/insertar un dato?

- ¿Cómo almacenamos los datos en cada servidor  $S_i$ ?



\* ¿Es esta solución más eficiente que una  $b+$ -tree en un solo servidor?  $\Rightarrow E(X_i) = \frac{1}{n}$

¿Cuál es el problema con esta solución?

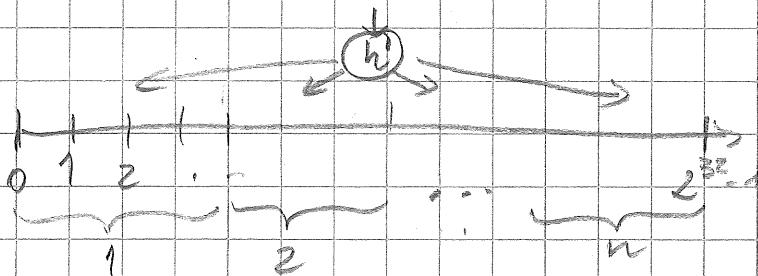
↳ inserción de una máquina n+1:  
 Requiere hacer re-hashing de todos los slots.

↳ ¿Por qué queremos insertar otras máquinas?

Hacer consistent hashing (otra solución)

Supongamos que en vez de tener una función de hash  $h: D \rightarrow \{1, \dots, n\}$ , tenemos una función  $h: D \rightarrow \{0, \dots, 2^{32}-1\}$  (palabas de 32-bit) y asignarlos a cada servidor si el conjunto

$$\{d \mid h(d) \in \left[ \frac{(k^2-1)}{n} \cdot (i-1), \frac{(k^2-1)}{n} \cdot i \right] \}$$



- Problema: - ¿Cómo borrar/insertar un dato?
- ¿Significa "uniforme" la distribución de los datos?
  - ¿Es posible invertir una máquina?
- Sin tener re-hashing de todos los datos.

### Consistent Hashing

Solución: Hacer hashing de datos y máquinas.

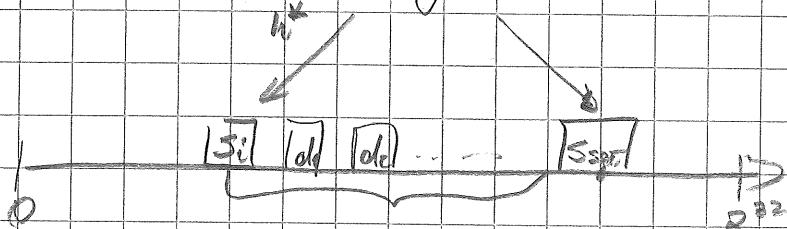
Supongamos que se tiene una función de hash:

$$h^*: D \times N \rightarrow \{0, \dots, 2^{32} - 1\}$$

y asignarlos a cada servidor si el conjunto

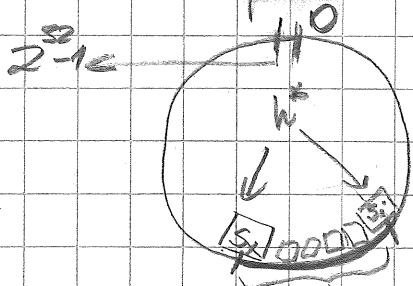
$$\{d | h^*(s_i) \leq h^*(d) < h^*(s_{soft}) \bmod 2^{32}\}$$

donde:  $s_{soft} = \arg \min_s h^*(h^*(s) - h^*(s_i)) \bmod 2^{32}$ .



Todos estos datos pertenecen a la máquina  $s_{soft}$ .

Esto también se puede ver como un circuito



→ todos estos estados  
son asignados a  $S_1$

Preguntas: - ¿Cómo busco / inserto un dato?

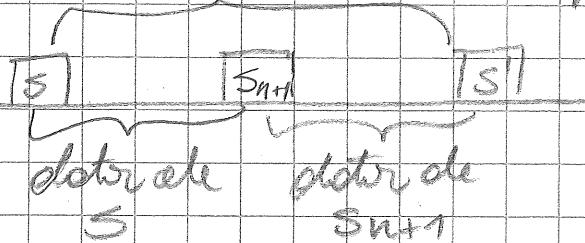
- Suposición: máquinas maestras (o todos los mag.)  
cuientan con un bit-tree de conjuntos

$$\{ h^*(s_i), s_i \}^3$$

ordenados por  $h^*(s_i)$ .

- ¿Cómo hacer para insertar una nueva  
máquina  $S_{n+1}$ ?

dato de  $S$  entre los adjacentes a  $S_{n+1}$



• - sub  $S$  debe mover sus datos de lugar

- Todos los otros elementos siguen iguales

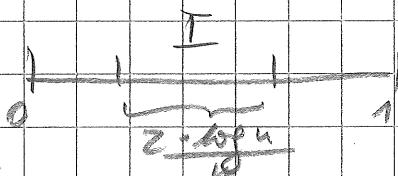
Pregunto: - ¿Es uniforme la corriente entre el interruptor?

Tercer punto: Con prob.  $1 - \frac{1}{n}$ , cada muestra es renovable de ésta más  $O(\log n/n)$  por los buckets del intervalo  $[0, 2^{32}-1]$ .

Dem: SP D6 supone que  $[0, 2^{32}-1]$  es el intervalo  $[0, 1]$  y sea:

$$S = \{h^*(s_i) \mid i \in \{1, \dots, n\}\}$$

Considera el subintervalo  $I$  de tam.  $\frac{2 \cdot \log n}{n}$



$$P(I \cap S = \emptyset) = \left(1 - \frac{2 \cdot \log n}{n}\right)^n = \left(\left(1 - \frac{2 \cdot \log n}{n}\right)^{\frac{n}{2 \cdot \log n}}\right)^{2 \cdot \log n}$$

$$\approx e^{-2 \cdot \log n} = \frac{1}{n^2}$$

igual

Subintervalos  $[0, 1]$   $I_1, \dots, I_k$  intervalos de tam  $\frac{2 \cdot \log n}{n}$

$$P(\text{alguno malo recibo}) \leq P\left(\bigcup_{i=1}^k I_i \cap S = \emptyset\right) \quad \text{y } k = \frac{n}{2 \cdot \log n}$$

más de  $\frac{4 \cdot \log n}{n}$  veces.

$$\leq \sum_{i=1}^k P(I_i \cap S = \emptyset) = \sum_{i=1}^k \frac{1}{n^2} \leq \frac{1}{n}$$

Note: esto supone que la función de hash distribuirá de manera uniforme los datos en el intervalo  $[0, 2^{3^k-1}]$  ( $(0, 1)$ ).

J) ¿Cómo podemos mejorar la uniformidad de la distribución en los servidores?

Considera que tiene una función  $h: D \rightarrow [0, 2^{3^k-1}]$

para los datos y  $h_i: N \rightarrow [0, 2^{3^k-1}]$   $k =$

función de hash para los servidores.

Entonces cada servidor  $S$  recibe los datos:

$$\{d \mid \exists i, h_i(s) \leq h(d) \} \quad \text{y} \quad N_i(s)$$

donde  $N_i(s) = \min_m \{h_i(s) \mid m = h_i(s)\}$ .

Preguntas: - j) cómo busco/inserto un valor?

↳ j) depende de  $k$ ?

- j) cómo insertamos una nueva muestra?

$$N_i(s) = \min \{h_j(s') \mid j \in k \wedge s' \in S, h_j(s) \geq h_i(s)\}$$

Tworacle: con prob  $1 - \frac{1}{2^n}$ , cada inserción es responsable de a lo más  $O(\log k \cdot n / n)$  buckets del intervalo  $[0, 2^{32} - 1]$

Dynamo, ejem., muy similar a la clm. anterior.

Presup: Suponge que cada inserción tiene una dependencia distintas.  
Como modificaciones consistent hashing pero que cada inserción lleva una dependencia "proporcional" a su copia?

— The end —

Comentarios:

- Aparece por primera vez en 1997
- Muy usado en sistemas P2P, BitTorrent, etc.
- Implementado por primera vez por Dynamo (amazon, 2006)

- Actualmente, la mayoría de los DB no SQL implementan una versión de consistent hashing como:
  - Couchbase, Cassandra, Voldemort, RIAK, ...
  - Es uno de los "trucos" dentro de DB no SQL.