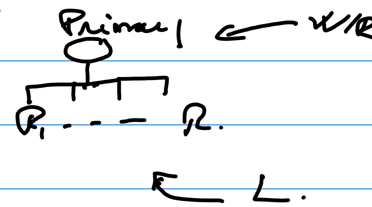
I Resumé

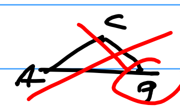
- BDR :
  - formidable : ACID
  - SQL : optim de Requêtes
  - limite : • Schema très rigide
    • ORM : Relat° - Impedance Mismatch.
    • Distribution.



- Distibution :

Theorème : CAP

|| coherence disponibilité



- key-value store.
  (redis).

Aujourd'hui

- Sharding / eclatement de la base
- Table de Hashage distribuée

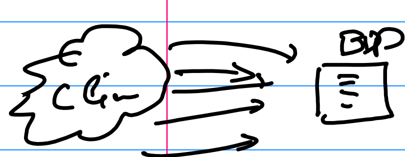
Elasticité : On a un système et on veut ajouter et enlever des Ressources.

II) Distribution des données.

- Distribution de la charge. (Primary Replica)

  vs

- distribut° des données (Eclatement / sharding)



trop de connexion => service interrompu

- BDD très grosse : – Primary / Replica va être coûteux
- BDD avec W/tH : – Primary / Replica peu efficace

On recherche une solution plus flexible :

- distribuable ac bcp de donnée de manière contrôlé
- distribué la charge en écriture
- et permettre la disponibilité

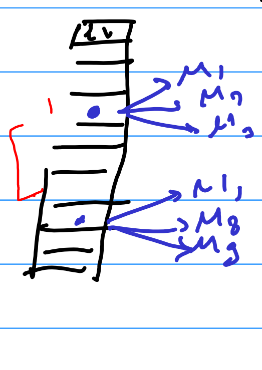
- Sharding / Éclatement :

On découpe les données en petit morceau (des éclats) qui vont être stocker chacun sur plusieurs machine.

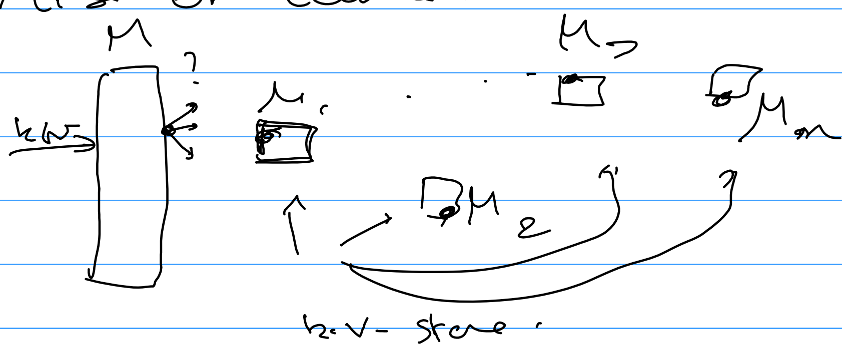
- Si chaque éclat est stocker sur k machine alors on parle d'un facteur de Replicat° de k.

⚠ Par la suite : un shard = une clés-valeur

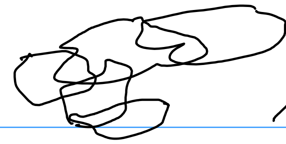
key values

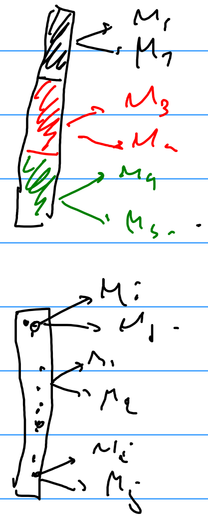


. Mise en œuvre :



Éclat très gros



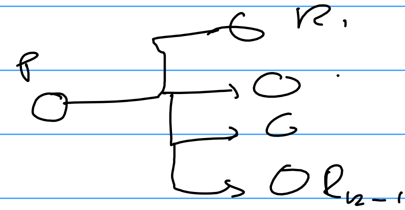


Éclat vossir

M_1, M_2 tombent $\Rightarrow$ $1/3$ de la base indisponible

M_1, M_2 tombent $\Rightarrow$ $\frac{1}{6 \times 5} = \frac{1}{30}$.

On reprend Primary Replica :



Sharding très gros qui contient toute la base.

III) Élasticité.

[Ajouter / Enlever à chaud des ressources au système

• hashage  $\Rightarrow$ redimensionner.

taux de remplissage important

(double la taille et on réinsert toutes les clés valeurs)

• Dans un mécanisme élastique

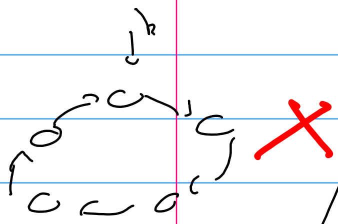
. On a plein de serveurs : $S_1, \dots, S_c$

. On veut savoir où stocker un k/v.

(k copie (taux de réplication))

• Le # de serveurs peut évoluer

• Cas simple : le nombre de serveurs est fixe.



Écrire :

• $k \to N$ on veut savoir où la stocker :

• Récupérer la valeur de k.

• $hash_1(k)$ [t]
$hash_2(k)$ [t]
$hash_i(k)$ [t]
} où j on stocke la clés.

• Redimensionnement nb de serveurs
⟹ on doit migrer quasi toute la base à chaque ajout

• Élasticité :
On ajoute un serveur ?

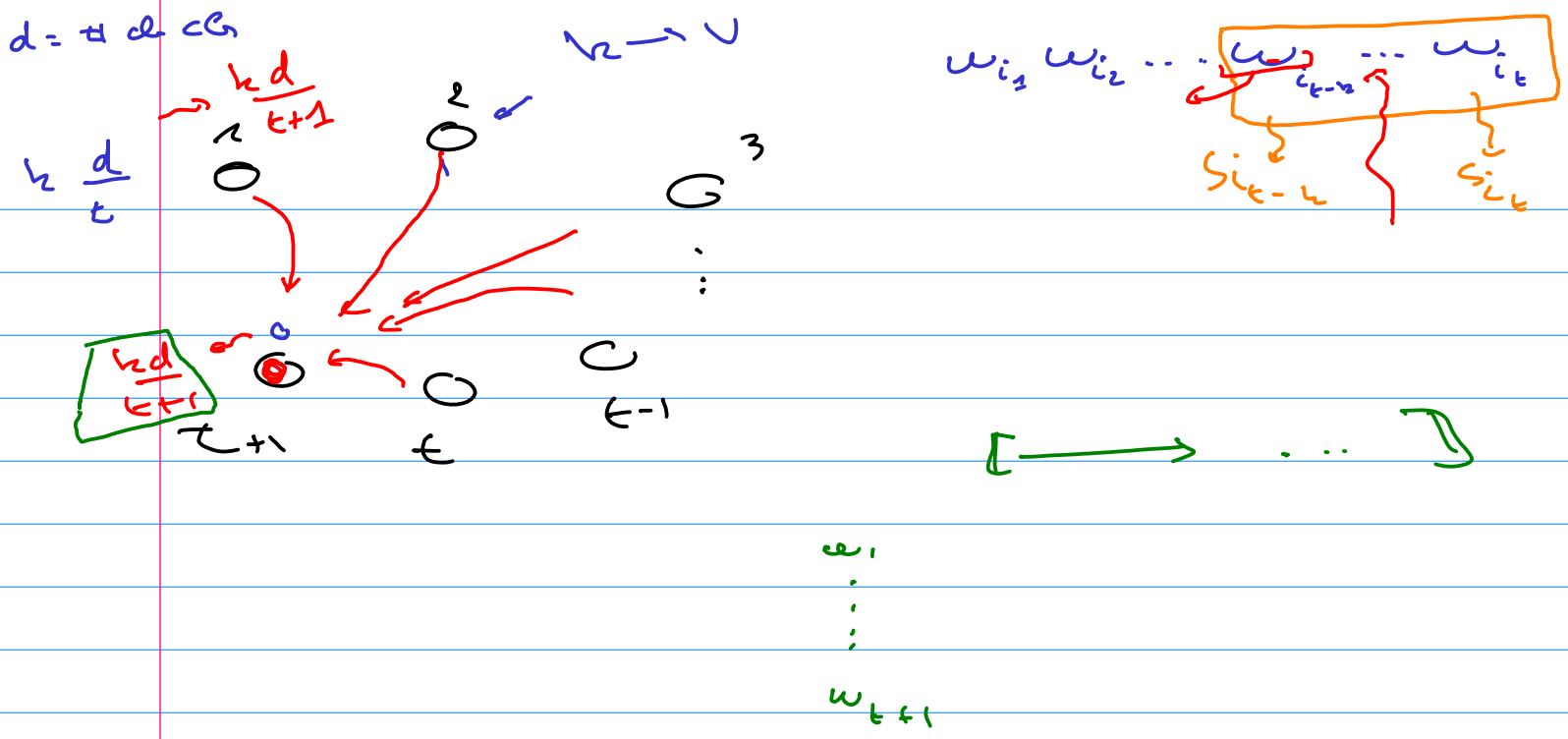
Rendez-vous hashing   Consistent hashing.

key → value

S_1 h_1 $h_1(key) = w_1 \in \mathbb{N}$
⋮ ⋮ ⋮ → poids
S_t h_t $h_t(key) = w_t \in \mathbb{N}$

On va choisir le k-serveur de poids maximal pour key

– Ajout d'un serveur $t \to t+1$
$\forall$ key de la base
On recalcule des k-serveurs par la même méthode et on migre les clefs



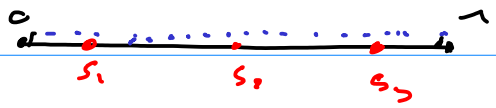
Consistent hashing.

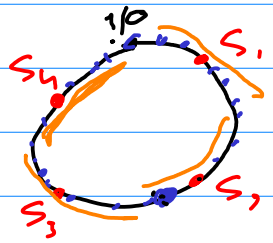
Servers + clés vont être "hashé" sur un même espace.

Rappel :

φ : Hash $\to [0,1[$

c_1





- Un envoie une clé sur le serveur dont elle est la plus proche

key → server $= \varphi(\text{hash}(\text{key}))$

on trouve le serveur le plus proche

$\text{argmin}_i \left| c_i - \varphi(\text{hash}(s)) \right|$

pour un taux de réplicat° de k
chaque clé va être hashé
k - fois.