Bases de données documentaires et distribuées, http://b3d.bdpedia.fr

Partitionnement

Vocabulaire

Rappel : une **partition** d'un ensemble S est un ensemble $\{F_1, F_2, \cdots F_n\}$ de parties de S telles que $F_i \cap F_j = \emptyset$ et $\bigcup_i Fi = S$.

Dans notre cas

- l'ensemble est une collection ;
- les éléments de l'ensemble sont les documents ;
- les parties sont nommées **fragments** (*shard* en anglais)

Le partitionnement (*sharding*)

Le partitionnement consiste à

- **découper** une (grande) collection en **fragments** en fonction d'une **clé**
- placer chaque fragment sur un **serveur**
- Maintenir un **répertoire** indiquant que telle **clé** se trouve dans tel **fragment** sur tel **serveur**

Le partitionnement apporte la **répartition de charge** (*load balancing*)

Il doit être **dynamique** (ajout/retrait de serveurs) pour s'adapter "élastiquement"

S'applique à des collections de paires (clé, valeur), où valeur est n'importe quelle information structurée.

La clé de partitionnement

La clé indique à quel fragment appartient un document.

Si un grand nombre de documents partagent la même clé, ils ne peuvent être que dans le même fragment $\Rightarrow$ le système perd en souplesse de distribution.

La valeur de la clé pour un document ne devrait pas être modifiable (nécessité de déplacement).

L'identifiant séquentiel d'un document est un bon choix

Opérations

Les opérations de type “dictionnaire” peuvent être transmises à **un seul** serveur :

- $get(k) : v$, recherche par clé
- $put(k, v)$, insertion
- $delete(k)$, suppression
- $range(k_1, k_2)$, recherche par intervalle (peut impliquer plusieurs serveurs, ne marche pas avec le hachage)

Pour toutes les autres opérations : tous les serveurs effectuent l'opération localement.

Exemple : recherche sur une clé secondaire, s'effectue sur **tous** les serveurs, avec si possible des index locaux.

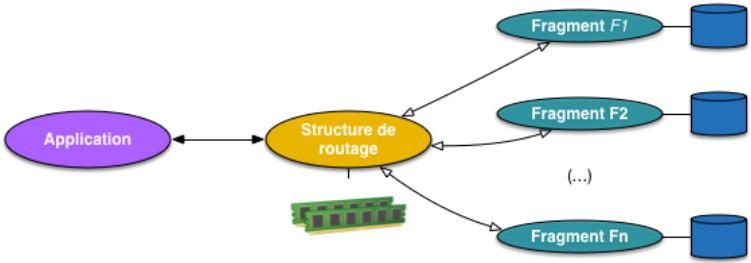
Techniques de partitionnement

- par **intervalle** : documents triés sur la clé, puis groupés par intervalles.
  Exemples représentatifs : MongoDB, HBase (BigTable).
- par **hachage** sur la clé, avec répertoire de hachage.
  Exemples : *memcached*, *chord*, *Dynamo/S3/Voldemort*, beaucoup d'autres.

Des méthodes connues dans d'autres contextes

Se référer aux techniques d'indexation par arbre-B et par hachage dans des environnements centralisés.

Structure d'un système partitionné

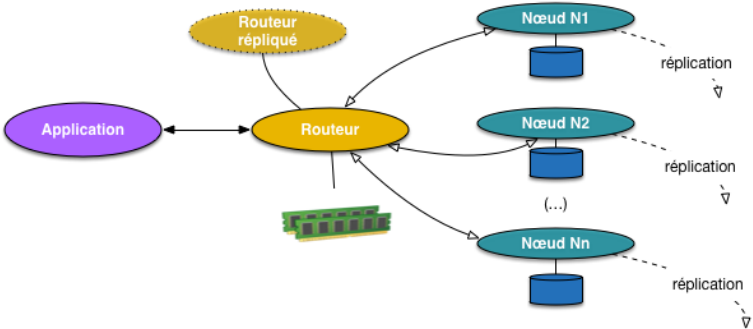


- Par intervalle = séquentiel indexé, **Arbre B**, Arbre B+, etc.
- Par hachage = hachage statique, dynamique, **hachage linéaire**.

Tout en RAM

La structure de routage doit tenir en mémoire pour être pleinement efficace.

Routage et stockage en distribué



- Routeur = *Master*, *Balancer*, *Config server*, ...
- Fragment = *chunk*, *tablet*, *region*, *bucket*, ...

Et la réplication ?

Chaque serveur gère une réplication locale pour tolérance aux pannes.