Stage ETE 2015

Description du système

F-PHT
Un système d'index de filtres de Bloom pour la recherche d'information par mots clés

Réalisé par **DOAN** Cao Sang
Encadrant : M. **MAKPANGOU** Mesaac, Regal

1 Juillet 2015

Table des matières

1 Vue globale **2**
1.1 Prefix Hash Tree (PHT) 2
1.2 F-PHT 4

2 Système d'indexation **5**
2.1 Objectif 5
2.2 Architecture du système d'indexation 5
2.3 API du système d'indexation 5

3 Fonctionnement du système **6**
3.1 Données de chaque nœud dans l'arbre 6
3.2 Protocole 6
3.2.1 Ajout d'un filtre de Bloom dans F-PHT 6
3.2.2 Recherche des filtres de Bloom dans F-PHT 6
3.2.3 Suppression d'un filtre de Bloom dans F-PHT 7

4 Implémentation du système **8**
4.1 Plateforme de simulation 8
4.2 Serveur 8
4.3 Implémentation du système 8

Chapitre 1

Vue globale

1.1 Prefix Hash Tree (PHT)

Un arbre préfixe est un arbre numérique ordonné qui est utilisé pour stocker une table associative où les clés sont généralement des chaînes de caractères. Contrairement à un arbre binaire de recherche, aucun nœud dans le trie ne stocke la chaîne à laquelle il est associé. C'est la position du nœud dans l'arbre qui détermine la chaîne correspondante [1].

Pour tout nœud, ses descendants ont en commun le même préfixe. La racine est associée à la chaîne vide. Des valeurs ne sont pas attribuées à chaque nœud, mais uniquement aux feuilles et à certains nœuds internes se trouvant à une position qui désigne l'intégralité d'une chaîne correspondante à une clé.

Pour faire une recherche d'une valeur associée à une clé, au départ, on se situe à la racine de l'arbre, en prenant le premier élément de la clé de la requête, on trouve le chemin étiqueté par cet élément, s'il n'existe pas, on est sûr que cette clé n'est pas dans l'arbre. Dès que l'on trouve le chemin, on arrive sur le bon nœud et continue en prenant le deuxième élément de la clé de requête, on applique cette méthode jusqu'à quand on trouve cette clé et se termine sur une feuille.

1. Wikipédia

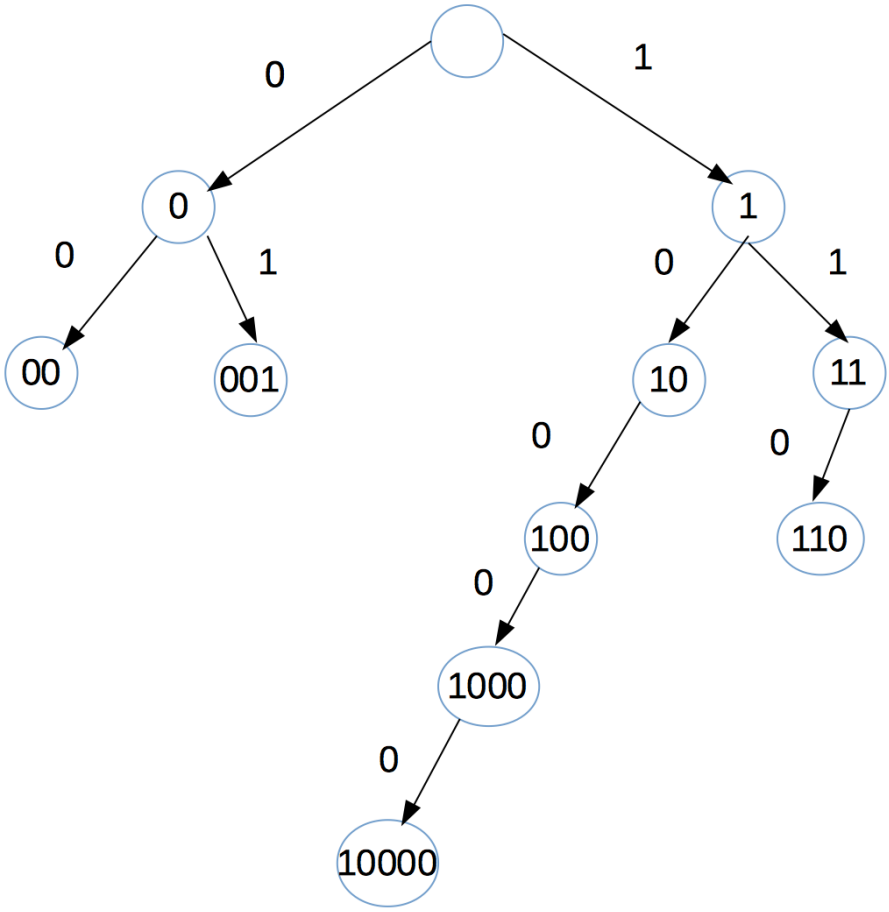

FIGURE 1.1 – Arbre de préfixe

1.2 F-PHT

F-PHT est un arbre préfixe de "multibit". Pour stocker une clé, on suit le chemin étiqueté par les fragments successifs de cette clé (dans l'ordre de leur rang) jusqu'à ce qu'on arrive sur une feuille pouvant stocker cette clé.

Un fragment est un morceau d'une clé (sous-ensemble de bits). Si on considère les clés de taille m, le système découpe chaque clé en f fragments de taille identique. Par convention, ces fragments sont numérotés de *0* à *f-1*.

15															0
1	0	0	0	1	1	0	1	0	0	0	0	1	0	1	0

TABLE 1.1 – Exemple le filtre de Bloom

La table 1.1 représente une clé de taille $m = 16$ bits, elle est ensuite découpée en $f = 4$ fragments. Le premier fragment est la suite de bits *"1000"* qui se trouve les plus à gauche de cette clé. Le dernier est *"1010"*.

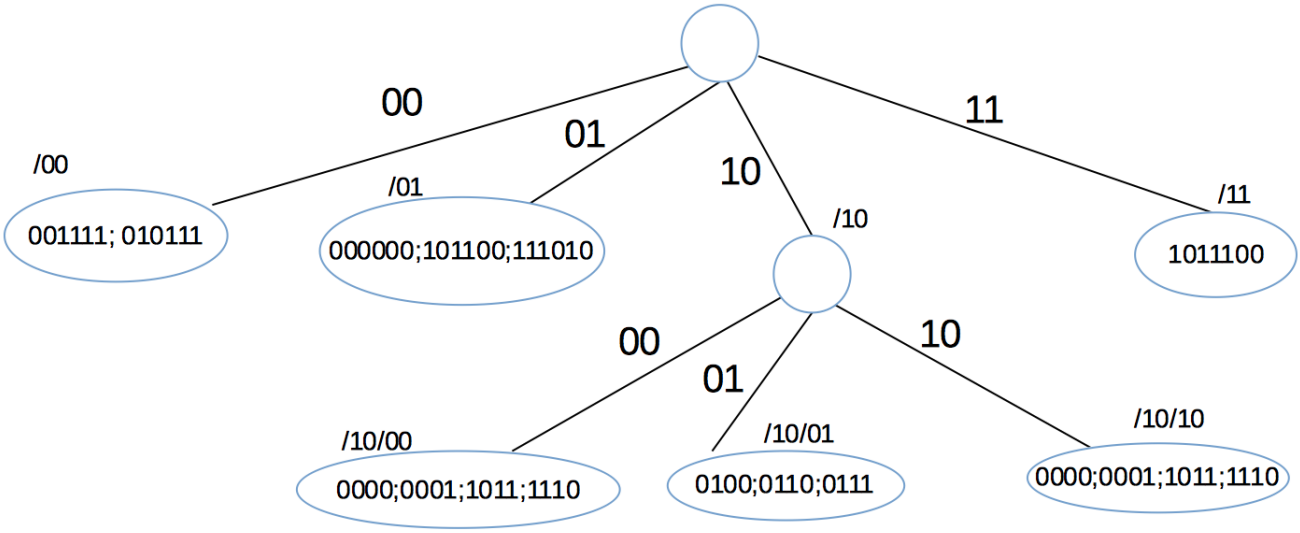

FIGURE 1.2 – Arbre F-PHT

La figure 1.2 représente l'arbre F-PHT qui stocke une clé de taille $m = 8$ bits, la taille d'un fragment est de 2 bits. Au dessus de chaque nœud (interne ou feuille) se trouve son identifiant unique et il caractérise également le chemin unique qui conduit vers ce nœud.

Chapitre 2

Système d'indexation

2.1 Objectif

Les clés indexées sont des filtres de Bloom de taille m. L'objectif du système est d'une part de stocker les filtres de Bloom donnés par les utilisateurs, d'autre part de permettre de rechercher tous les filtres de Bloom (stockés dans le système) qui sont sur-ensemble d'un filtre de Bloom caractérisant des mots clés d'une requête.

2.2 Architecture du système d'indexation

Le système d'indexation est réalisé par un ensemble de serveurs d'index répartis sur plusieurs machines. Chaque serveur gère un ou plusieurs nœuds de l'arbre F-PHT. Ce système permet de créer plusieurs index, chaque index maintient un F-PHT qui stocke les clés.

On suppose qu'il y a un serveur central connu qui gère tous les index créés. Il utilise une table des index, chaque entrée sert à sauvegarder l'identifiant de l'index *indexID* et le serveur *rootNodeID* qui héberge sa racine. Lorsqu'un utilisateur veut créer/supprimer un index, il demande le serveur central d'ajout/suppression une entrée dans la table des index.

2.3 API du système d'indexation

Ce système offre aux utilisateurs des services suivants :

createIndex() : créer un nouveau index et rend aux l'utilisateur l'identifiant d'index créé *indexID*.

removeIndex(indexID) : supprime un index identifié par *indexID*.

add(indexID, key) : ajoute dans l'index identifié par *indexID* une clé *key*.

remove(indexID, key) : supprime une clé *key* dans l'index identifié par *indexID*.

search(indexID, key_req) : recherche dans l'index identifié par *indexID* des sur-ensembles qui satisfont la requête *key_req*.

Chapitre 3

Fonctionnement du système

3.1 Données de chaque nœud dans l'arbre

Identifiant d'un nœud caractérise le nœud. Cet identifiant est calculé grâce au chemin vers ce nœud, et ce chemin est unique.

Niveau d'un nœud (rang) est l'hauteur dans l'arbre F-PHT.

Table de routage (LocalRoute) sert :
- soit à router chaque requête vers le nœud fils responsable de la traiter.
- soit à déterminer l'adresse du conteneur local. Ce conteneur stocke au maximum γ éléments.

3.2 Protocole

3.2.1 Ajout d'un filtre de Bloom dans F-PHT

Lorsqu'un nœud reçoit une requête d'ajout d'un filtre de Bloom (c'est-à-dire d'une clé), il effectue deux actions :

1. Premièrement, il détermine la valeur du fragment numéro i de la clé à ajoutée, avec i correspondant au niveau de ce nœud dans l'arbre F-PHT.
2. Une fois le fragment de la clé de rang i déterminé, il consulte l'entrée de la table de routage *LocalRoute* associée à cette valeur. Trois cas sont à distinguer :
   (a) Si l'entrée est nulle (aucune clé ayant la même valeur pour le fragment de rang i a été ajoutée auparavant), le nœud crée un conteneur local, ajoute son adresse dans cette entrée et stocke la clé reçue dans ce conteneur.
   (b) Si l'entrée associée à la valeur du fragment de rang i contient l'identifiant d'un nœud fils, on transmet la requête vers ce fils.
   (c) Si l'entrée contient l'adresse d'un conteneur local, on ajoute la clé reçue dans ce conteneur. Si après ajout, le nombre d'éléments stockés dépasse le seuil fixé, le nœud crée un nœud fils, lui transmet le conteneur, et met à jour l'entrée correspondante l'adresse de son nouveau fils.

3.2.2 Recherche des filtres de Bloom dans F-PHT

Lorsqu'un nœud reçoit une requête de recherche des sur-ensembles d'un filtre de Bloom, il effectue les étapes suivantes :

1. D'abord, il détermine la valeur du fragment numéro i de la clé de requête, avec i correspondant au niveau de ce nœud dans l'arbre F-PHT.

2. Ensuite, il consulte toutes les entrées de la table de routage *LocalRoute* contiennent la valeur du fragment déterminée précédente. Pour chaque entrée, il y a 3 cas à traiter :
   (a) Si l'entrée est nulle (aucune clé contenant cette valeur du fragment de rang i), le nœud continue.
   (b) Si l'entrée associée à la valeur du fragment de rang i contient l'identifiant d'un nœud fils, on transmet la requête vers ce fils.
   (c) Si l'entrée contient l'adresse d'un conteneur local, on collecte tous les filtres qui contiennent la requête. L'ensemble des filtres collectés est retourné comme réponse.
3. Puis, il attend toutes les réponses des fils auxquels il ont transmis la requête. Une fois, il reçoit toutes les réponses et réunit avec son ensemble des résultats, il transmet le résultat au père.

3.2.3 Suppression d'un filtre de Bloom dans F-PHT

Les étapes de la suppression d'un filtre de Bloom dans F-PHT effectuées par un nœud sont :

1. Il détermine la valeur du fragment numéro i de la clé de requête, avec i correspondant au niveau de ce nœud dans l'arbre F-PHT.
2. Ensuite, il consulte l'entrée de la table de routage *LocalRoute* associée à cette valeur. Trois cas sont à distinguer :
   (a) Si l'entrée est null (aucune clé ayant la même valeur pour le frament numéro i a été ajoutée auparavant), le nœud rejette la requête.
   (b) Si l'entrée associée à la valeur du fragment de rang i contient l'identifiant d'un nœud fils, on transmet la requête vers ce fils.
   (c) Si l'entrée contient l'adresse d'un conteneur local, on supprime la clé de requête dans ce conteneur. Si après suppression, ce conteneur devient vide, on supprime ce conteneur et cette entrée dans la table *LocalRoute*. Si cette table devient vide, il notifie le père de la suppression de son fils et demande le système de supprimer ce nœud. Si le père reçoit la notification de son fils, il vide l'entrée où stocke son adresse. En plus, si la table de routage *LocalRoute* de son père devient aussi vide, il fait la même procédure comme son fils : notifie son père, demande la suppression du système.

Chapitre 4

Implémentation du système

4.1 Plateforme de simulation

Cette plateforme est destinée à simuler le système d'indexation, dans un premier temps, il est centralisé sur une seule machine. Elle crée un index, ensuite ajoute des données dans cet index. A la fin, une suite de requête de recherche des données est lancée pour tester le fonctionnement du système. Elle peut également simuler la suppression des données dans le système.

Le simulateur maintient une table des des instances de serveurs existants dans ce système, il joue aussi le serveur de localisation des index.

L'application interagit avec le système via l'API.

4.2 Serveur

4.3 Implémentation du système