

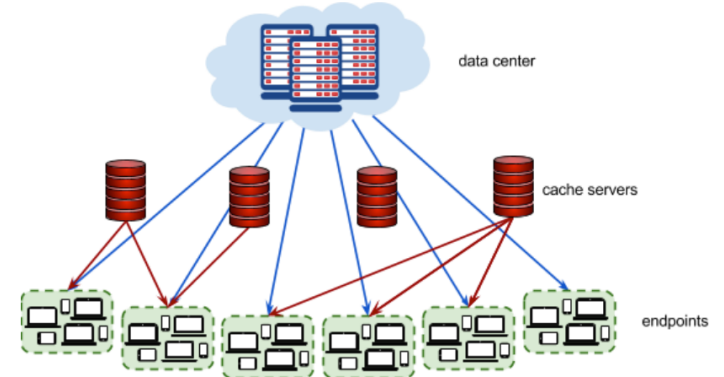
Google Hash Code

Streaming videos

Hash Code 2017, Online Qualification Round

Énoncé

Représentation du problème



Énoncé

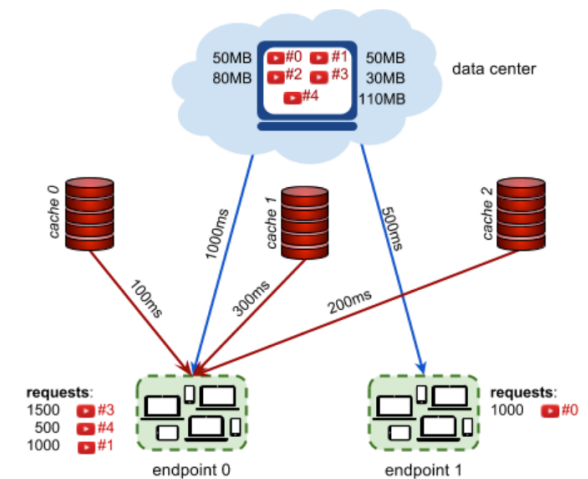
Représentation du problème

- d : le data center
- V : les vidéos
 - $\forall v \in V, s_v$: la taille en MB de la vidéo
- C : les cache servers, avec une capacité en MB
- E : les endpoints, groupes d'utilisateurs
 - $\forall e \in E, L_{e,d}$: la latence entre e et le data center
 - $\forall e \in E, L_{e,c}$: la latence entre e et le cache server c
- R : les requêtes
 - $r_{e,v}$: le nombre de requêtes pour la vidéo v de l'endpoint e

Objectif : Minimiser le temps d'attente moyen pour toutes les requêtes

Exemple

Exemple



Exemple

Exemple

- 5 vidéos
 - $s_0 = 50, s_1 = 50, s_2 = 80, s_3 = 30, s_4 = 110$
- 3 cache servers, avec une capacité de 100MB
- 2 endpoints
 - $L_{0,d} = 1000, L_{0,0} = 100, L_{0,2} = 200, L_{0,1} = 300$
 - $L_{1,d} = 500$
- 4 requêtes
 - $r_{0,3} = 1500, r_{0,4} = 500, r_{0,1} = 1000$
 - $r_{1,0} = 1000$

Énoncé

Les variables ?

- Les videos dans les cache servers
 - $\forall c \in C, l_c$: la liste des vidéos dans le cache server c

Algorithme glouton

Principe

- À chaque étape, on fait un choix, celui qui semble le meilleur à cet instant
- Construit une solution pas à pas
 - sans revenir sur ses décisions
 - en effectuant à chaque étape le choix qui semble le meilleur
 - en espérant obtenir un résultat optimum global
- Approche glouton
 - suivant les problèmes pas de garantie d'optimalité (heuristique gloutonne)
 - peu coûteuse (comparée à une énumération exhaustive)
 - choix intuitif

Algorithme glouton

Exemple

- 3 cache servers, avec une capacité de 100 MB
- $s_0 = 50, s_1 = 50, s_2 = 80, s_3 = 30, s_4 = 110$
- $L_{0,d} = 1000, L_{0,0} = 100, L_{0,1} = 300, L_{0,2} = 200, L_{1,d} = 500$
- $r_{0,3} = 1500, r_{0,4} = 500, r_{0,1} = 1000, r_{1,0} = 1000$

Objectif : Minimiser le temps d'attente moyen pour toutes les requêtes

- On trie les requêtes par ordre décroissant
- On parcourt les requêtes et on essaye de placer la vidéo de manière à minimiser le temps de latence

Algorithme glouton

Exemple

- 3 cache servers, avec une capacité de 100 MB
- $s_0 = 50, s_1 = 50, s_2 = 80, s_3 = 30, s_4 = 110$
- $L_{0,d} = 1000, L_{0,0} = 100, L_{0,1} = 300, L_{0,2} = 200, L_{1,d} = 500$
- $r_{0,3} = 1500, r_{0,1} = 1000, r_{1,0} = 1000, r_{0,4} = 500$

- $l_0 = [3, 1]$ capacité restante 20MB
- $l_1 = []$ capacité restante 100MB
- $l_2 = []$ capacité restante 100MB

- $1500 \times 100 + 1000 \times 100 + 1000 \times 500 + 500 \times 1000 = 1250000\text{ms}$

Algorithme glouton

Exemple

- 3 cache servers, avec une capacité de 100 MB
- $s_0 = 50, s_1 = 50, s_2 = 80, s_3 = 30, s_4 = 110$
- $L_{0,d} = 1000, L_{0,0} = 100, L_{0,1} = 300, L_{0,2} = 200, L_{1,d} = 500, L_{1,0} = 100$
- $r_{0,3} = 1500, r_{0,1} = 1000, r_{1,0} = 1000, r_{0,4} = 500$

- $l_0 = [3, 0]$ capacité restante 20MB
- $l_1 = []$ capacité restante 100MB
- $l_2 = [1]$ capacité restante 50MB

- 950000ms

Algorithme glouton

Exemple

- 3 cache servers, avec une capacité de 100 MB
- $s_0 = 50, s_1 = 50, s_2 = 80, s_3 = 30, s_4 = 110$
- $L_{0,d} = 1000, L_{0,0} = 100, L_{0,1} = 300, L_{0,2} = 200, L_{1,d} = 500$
- $r_{1,0} = 1000, r_{0,3} = 1500, r_{0,1} = 1000, r_{0,4} = 500$

Amélioration

On peut changer de stratégie

- 1 On trie les requêtes par ordre décroissant
- 2 On trie les requêtes par **regret** (différence entre les 2 plus petites latences)
- 3 Une combinaison des 2 précédentes