Placement de données optimal pour des réseaux en ligne et en anneau

Eric Angel[1], Evripidis Bampis[1], Fadi Kacem[1], Vassilis Zissimopoulos[2]

[1] IBISC-CNRS FRE 3190 ; Université d'Evry Val d'Essonne ; Boulevard Mitterrand, 91025 Evry, France
{angel,bampis,fkacem}@ibisc.fr
[2] Department of Informatics and Telecommunications ; University of Athens, Greece
vassilis@di.uoa.gr

Mots-Clés : *allocation de données, programmation dynamique, ligne, anneau.*

Introduction. Les réseaux de partage de fichiers pair-à-pair sont devenus un aspect très populaire de l'utilisation quotidienne de l'internet. Le succès de tels systèmes est basé sur l'exploitation d'une nouvelle ressource, le stockage distribué. L'utilisation massive de cette nouvelle ressource est dûe au fait que des capacités de stockage plus importantes sont actuellement disponibles avec un coût très faible et avec des temps d'accès très rapides. Ainsi les utilisateurs intéragissent en installant un stockage local (cache), en répliquant les contenus les plus populaires et en les mettant à disposition des utilisateurs du voisinage, diminuant de cette manière la consommation de bande passante.

Le problème du *placement de données* (objets) offre un modèle abstrait approprié pour modéliser cette situation : un ensemble de clients (machines ou utilisateurs) reliés selon une certaine topologie est considéré, chaque client disposant d'une capacité de stockage donnée. Etant donnés l'ensemble d'objets disponibles et la préférence de chaque client pour chaque objet, l'objectif est de proposer un schéma de réplication d'objets, autrement dit un placement des objets (et de leurs copies) sur les machines minimisant le coût total d'accès des clients aux objets.

Le problème du placement de données a été largement étudié dans la littérature. Dans [1], les auteurs ont montré que le problème est MAXSNP-difficile et ils ont proposé un algorithme 20.5-approché dans le cas général. Ce résultat a été amélioré dans [2] où un algorithme 10-approché a été proposé. Plus récemment, un algorithme optimal a été proposé dans [3] dans le cas où tous les objets sont uniformes et le nombre de clients est borné par une constante. D'autres résultats existent lorsqu'on permet un dépassement de la capacité des clients. Ici, nous proposons un algorithme optimal, basé sur la programmation dynamique, dans le cas où la topologie du réseau est une ligne et sans violation de la capacité des clients supposée bornée. Nous étendons ce résultat lorsque la topologie est un anneau. Ces deux résultats sont valides dans le cas où le nombre de clients est une entrée du problème (i.e. non borné par une constante).

Présentation du problème. On considère un réseau constitué d'un ensemble $\mathcal{M}$ de M utilisateurs, reliés entre eux selon une certaine topologie $\mathcal{N}$, ainsi qu'un ensemble $\mathcal{O}$ de N objets de taille unitaire qui peuvent être répliqués.
Chaque utilisateur i demande à accéder à un ensemble $R_i \subseteq \mathcal{O}$ d'objets selon une certaine fréquence. On note w_{io} la fréquence d'accès de l'objet $o \in R_i$ pour l'utilisateur i. Considèrons un objet particulier $o \in R_i$. Si l'utilisateur i possède l'objet, alors son coût d'accès est égal à zéro. Sinon, si l'objet o est stocké dans le cache d'un utilisateur voisin j (selon la topologie $\mathcal{N}$), alors le coût d'accès est défini comme étant égal à $d_{ji}w_{io}$, avec d_{ji} la distance entre i et j. Si l'objet o n'est pas stocké localement,

$\mathcal{P}_0 = \{P_0\}$, $\mathcal{P}_{M+1} = \{P_{M+1}\}$ avec $P_0 = P_{M+1} = \emptyset$.
Pour chaque $P_{M-1} \in \mathcal{P}_{M-1}$
  Pour chaque $P_M \in \mathcal{P}_M$
    $C^*_M(P_{M-1}, P_M) := C_M(P_{M-1}, P_M, P_{M+1})$
Pour $k = M-1$ **jusqu'à** 1 **par pas de** -1
  Pour chaque $P_{k-1} \in \mathcal{P}_{k-1}$
    Pour chaque $P_k \in \mathcal{P}_k$
      $C^*_k(P_{k-1}, P_k) := min_{P_{k+1} \in \mathcal{P}_{k+1}}$
        $\{C_k(P_{k-1}, P_k, P_{k+1}) + C^*_{k+1}(P_k, P_{k+1})\}$
Retourner $\min_{P_1 \in \mathcal{P}_1} C^*_1(P_0, P_1)$.

Pour chaque $P_M \in \mathcal{P}_M$
  Pour chaque $P_1 \in \mathcal{P}_1$
    Pour chaque $P_{M-1} \in \mathcal{P}_{M-1}$
      $C^*_M(P_{M-1}, P_M) := C_M(P_{M-1}, P_M, P_1)$
    Pour $k = M-1$ **jusqu'à** 3 **par pas de** -1
      Pour chaque $P_{k-1} \in \mathcal{P}_{k-1}$
        Pour chaque $P_k \in \mathcal{P}_k$
          $C^*_k(P_{k-1}, P_k) := min_{P_{k+1} \in \mathcal{P}_{k+1}}$
            $\{C_k(P_{k-1}, P_k, P_{k+1}) + C^*_{k+1}(P_k, P_{k+1})\}$
    Pour chaque $P_2 \in \mathcal{P}_2$
      $C^*_2(P_1, P_2) := min_{P_3 \in \mathcal{P}_3}\{C_2(P_1, P_2, P_3) + C^*_3(P_2, P_3)\}$
    $OPT(P_M, P_1) := \min_{P_2 \in \mathcal{P}_2}\{C_1(P_M, P_1, P_2) + C^*_2(P_1, P_2)\}$
Retourner $\min_{P_M \in \mathcal{P}_M, P_1 \in \mathcal{P}_1} OPT(P_M, P_1)$

FIG. 1 – Algorithmes de programmation dynamique pour les lignes et les anneaux.

ni dans le cache d'un utilisateur voisin, l'utilisateur i doit faire appel à un serveur distant s (de capacité non bornée) avec un coût d'accès $d_{si} \cdot w_{io}$, où d_{si} est une constante (qui peut être égale à $+\infty$). Chaque utilisateur $i \in \mathcal{M}$ peut stocker au maximum C_i objets dans son cache. L'objectif est de déterminer un placement optimal minimisant le coût total d'accès des utilisateurs aux objets.

Algorithme de programmation dynamique. On note P_i un placement d'objets sur l'utilisateur i, i.e P_i est un sous-ensemble d'objets qu'on choisit de répliquer localement. Pour simplifier la présentation, nous introduisons deux utilisateurs virtuels 0 et $M+1$ avec des placements vides sur chacun d'eux, i.e. $P_0 = P_{M+1} = \emptyset$, et nous supposons que les utilisateurs sont répartis le long d'une ligne, avec l'utilisateur i ($1 \leq i \leq M$) voisin des utilisateurs $i-1$ et $i+1$.

On note $\mathcal{P}_i$ l'ensemble de tous les placements d'objets réalisables sur l'utilisateur i. Nous avons $|\mathcal{P}_i| \leq N^{C_i} \leq N^C$, avec $C = max_{i \in \mathcal{M}}(C_i)$. Pour $1 \leq k \leq M$, nous notons $C_k(P_{k-1}, P_k, P_{k+1})$ le coût d'accès induit par le placement P_k sur l'utilisateur k, quand ses utilisateurs voisins $k-1$ et $k+1$ ont reçu des placements P_{k-1} et P_{k+1} respectivement. Nous avons $C_k(P_{k-1}, P_k, P_{k+1}) = \sum_{\substack{o \in R_k \cap P_{k-1} \cap P_{k+1} \\ o \notin P_k}} \min\{d_{k-1,k}, d_{k+1,k}\}\, w_{ko} + \sum_{\substack{o \in R_k \cap P_{k-1} \\ o \notin P_k \cup P_{k+1}}} d_{k-1,k} w_{ko} + \sum_{\substack{o \in R_k \cap P_{k+1} \\ o \notin P_k \cup P_{k-1}}} d_{k+1,k} w_{ko} + \sum_{\substack{o \in R_k \\ o \notin P_k \cup P_{k-1} \cup P_{k+1}}} d_{sk}\, w_{ko}$. Pour $k \in \{1, \ldots, M\}$, étant donné le placement P_{k-1} (resp. P_k) sur l'utilisateur $k-1$ (resp. k), nous notons $C^*_k(P_{k-1}, P_k)$ le coût d'une solution optimale minimisant la somme des coûts d'accès des utilisateurs $k, k+1, \ldots$ jusqu'à M. Le coût total optimal recherché est alors donné par $\min_{P_1 \in \mathcal{P}_1} C^*_1(P_0, P_1)$. Nous avons $C^*_k(P_{k-1}, P_k) = min_{P_{k+1} \in \mathcal{P}_{k+1}}\{C_k(P_{k-1}, P_k, P_{k+1}) + C^*_{k+1}(P_k, P_{k+1})\}$ pour $k \in \{1, \ldots, M-1\}$, et $C^*_M(P_{M-1}, P_M) = C_M(P_{M-1}, P_M, P_{M+1})$. À partir de ces relations il est possible d'en déduire un algorithme de programmation dynamique, donné dans la figure 1 (à gauche), de complexité $\mathcal{O}(MN^{3C})$. De manière similaire, il est possible d'obtenir un algorithme de programmation dynamique pour une topologie en anneau. On note $OPT(P_M, P_1)$ le coût d'une solution optimale sous réserve d'avoir choisi le placement P_M (resp. P_1) pour l'utilisateur M (resp. 1). L'algorithme est donné dans la figure 1 (à droite), et est de complexité $\mathcal{O}(MN^{4C})$.

Références

[1] Ivan D. Baev and Rajmohan Rajaraman. *Approximation algorithms for data placement in arbitrary networks.* In Proceedings of the ACM-SIAM Annual Symposium on Discrete Algorithms (SODA), pages 661-670, 2001.

[2] Ivan D. Baev, Rajmohan Rajaraman, and Chaitanya Swamy. Approximation algorithms for data place- ment problems. SIAM Journal on Computing, 38(4) :1411-1429, 2008.

[3] Eric Angel, Evripidis Bampis, Gerasimos G. Poliatos, Vassilis Zissimopoulos. *Optimal data placement on networks with constant number of clents*, soumis, 2009.