Fouille de données dans les systèmes Pair-à-Pair pour améliorer la recherche de ressources

Florent Masseglia*, Pascal Poncelet**, Maguelonne Teisseire***

*INRIA Sophia Antipolis, Axis Project-Team, BP93 06802 Sophia Antipolis - France
Florent.Masseglia@sophia.inria.fr
**EMA-LGI2P/Site EERIE, Parc Scientifique Georges Besse, 30035 Nîmes Cedex, France
{Pascal.Poncelet}@ema.fr
***LIRMM UMR CNRS 5506, 161 Rue Ada, 34392 Montpellier cedex 5 - France
{teisseire}@lirmm.fr

Résumé. La quantité de sources d'information disponible sur Internet fait des systèmes d'échanges pair-à-pair (P2P) un genre nouveau d'architecture qui offre à une large communauté des applications pour partager des fichiers, des calculs, dialoguer ou communiquer en temps réel. Dans cet article, nous proposons une nouvelle approche pour améliorer la localisation d'une ressource sur un réseau P2P non structuré. En utilisant une nouvelle heuristique, nous proposons d'extraire des motifs qui apparaissent dans un grand nombre de noeuds du réseau. Cette connaissance est très utile pour proposer aux utilisateurs des fichiers souvent demandés (en requête ou en téléchargement) et éviter une trop grande consommation de la bande passante.

1 Introduction

La quantité de sources d'information disponible sur Internet fait des systèmes d'échanges pair-à-pair (P2P) un genre nouveau d'architecture qui offre à une large communauté des applications pour partager des fichiers, partager des calculs, dialoguer ou communiquer en temps réel, etc (Miller (2001), Ngan et al. (2003)). Les applications P2P fournissent également une bonne infrastructure pour les opérations sur de grandes masses de données ou avec de très nombreux calculs, comme la fouille de données. Dans ce cadre, nous considérons une nouvelle approche pour améliorer la localisation de ressources dans un environnement P2P non structuré selon deux aspects principaux pour extraire des comportements fréquents :

1. *L'ordre des séquences* entre les actions réalisées sur les nœuds (requête ou téléchargement) est pris en compte pour améliorer les résultats.
2. Les résultats des calculs distribués sont maintenus via un *"Pair centralisé"* pour réduire le nombre de communications entre les pairs connectés.

Connaître l'ordre des séquences des actions réalisées sur les pairs offre une connaissance importante. Par exemple, en examinant les actions réalisées, nous pouvons savoir que pour 77% des nœuds pour lesquels il y a une requête concernant "*Mandriva Linux*", le fichier "*Mandriva Linux 2005 CD1 i585-Limited-Edition-Mini.iso*" est choisi et téléchargé. Cette requête

est suivie par la demande des images iso (*i.e* "*Mandriva Linux 2005 Limited Edition*"), et dans la grande quantité de résultats retournés, l'image "*Mandriva Linux 2005 CD2 i585-Limited-Edition-Mini.iso*" est choisie et téléchargée. L'un des problèmes principaux des systèmes P2P non structurés comme Gnutella est que les requêtes sont envoyées à un trop grand nombre de nœuds (*broadcast*) entraînant ainsi une consommation excessive de la bande passante (Ng et al. (2003)). Proposer à l'avance à l'utilisateur les fichiers souvent associés à une requête ou à un téléchargement permet d'éviter une consommation excessive de la bande passante dans la mesure où l'on connaît à l'avance les ressources à extraire. Il suffit alors d'enrichir le résultat de la première requête avec des informations complémentaires sur les fichiers majoritairement téléchargés par les autres utilisateurs.

Rechercher des règles d'association ou des motifs séquentiels dans un système aussi distribué que les systèmes P2P n'est pas une tâche facile. En effet, par nature, ces systèmes sont très dynamiques, i.e. les nœuds agissent indépendamment les uns des autres, et les connaissances acquises ne sont alors plus forcément représentatives. Par exemple, quand un nœud disparaît, les séquences de ce nœud disparaissent également de la base distribuée et la connaissance extraite doit être reconsidérée. Les approches d'extraction de motifs séquentiels traditionnelles (Srikant (1995), Pei et al. (2001)) qui considèrent que la base est disponible dans son intégralité ne sont donc plus utilisable dans un contexte aussi dynamique. Notre proposition se situe dans ce cadre et prend en considération l'aspect dynamique des systèmes pairs à pairs non structurés.

Dans la suite de cet article, la section 2 présente la problématique de la recherche de motifs séquentiels dans une base de données distribuée. Dans la section 3, nous proposons une nouvelle approche basée sur une heuristique. Les expérimentations menées sont décrites dans la section 4. Puis la section 5 conclut cet article.

2 Problématique

Dans cette section, nous étendons la problématique intitiale de la recherche de motifs séquentiels Srikant (1995) à un environnement P2P non structuré. Soit $I = \{x_1, \ldots, x_n\}$ un ensemble de littéraux distincts appelés items. Nous considérons par la suite que pour chaque item nous connaissons l'action réalisée, i.e. requête ou téléchargement. Un item x_i est donc noté soit $[d, x_i]$ pour une action de téléchargement ou $[r, x_i]$ pour une requête. Une *séquence* est une liste ordonnée d'itemsets notée par $< s_1 s_2 \ldots s_n >$ où s_j est un itemset. Par exemple, considérons les actions réalisées sur un nœud S=$< ([d, 3]) \ ([d, 4] \ [d, 5]) \ ([d, 8]) >$. S s'interprète de la manière suivante : 3 a d'abord été téléchargé, puis 4 et 5 au même moment (i.e. dans la même transaction) et finalement 8.

Soit D_t la base au temps t, dans un système P2P non structuré, cette base est répartie sur différents nœuds. Pour un nœud u, nous notons sa partition au temps t par D_t^u. En fait, D_t^u correspond aux séquences d'actions réalisées dans le nœud u au temps t. Soit $D_t = \bigcup D_t^u$ ... D_t^v, où u_t ... v_t sont les nœuds disponibles au temps t. Le problème de la recherche de motifs séquentiels dans un tel environnement distribué consiste à rechercher les séquences fréquentes dans D_t en fonction d'une valeur de support minimal $minsupp$, i.e. les séquences dont le nombre d'occurrences dans D_t est supérieur ou égal à $minsupp$.

3 Une nouvelle heuristique pour extraire des motifs séquentiels dans des systèmes pair-à-pair non structurés

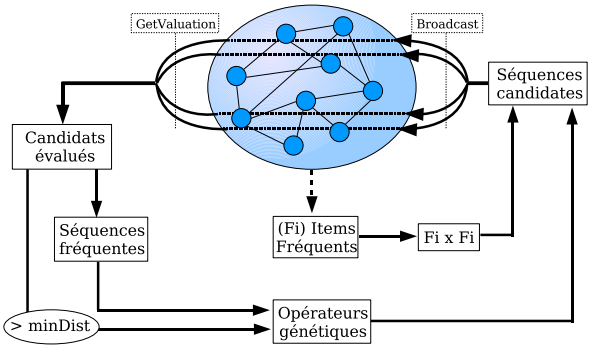

FIG. 1 – *Aperçu de l'heuristique* $Distributed_{SP}$

Les nœuds dans un système P2P peuvent se connecter ou disparaître alors que D_t est en cours d'analyse. Notre proposition considère que D_t est capable de recevoir des séquences candidates, d'évaluer leur support et de retourner le résultat. Ce type de "parcours" distribué sur tous les nœuds tire profit d'algorithmes stochastiques. Tout d'abord D_t est vide jusqu'à ce qu'un nœud envoie sa séquence. L'architecture P2P non structuré que nous proposons utilise un pair spécial (appelé par la suite pair "$Distributed_{SP}$") qui est connecté à tous les nouveaux pairs qui arrivent sur le réseau (l'instruction *send(*v,@$Distributed_{SP}$*)* dans l'algorithme 2, permet à $Distributed_{SP}$ d'être au courant de l'arrivée du nœud "v"). Notre méthode utilise alors une distribution des séquences candidates comme illustrée par la figure 1. Le pair "$Distributed_{SP}$" réalise les instructions de l'heuristique $Distributed_{SP}$, de "GetValuation" à "Broadcast". Tout d'abord, l'ensemble des items fréquents est extrait des pairs connectés. Puis l'ensemble de tous les candidats de taille 2 est généré. Ces candidats sont évalués par les pairs connectés ($u_t..v_t$) pour connaître ceux qui ont un nombre d'occurrences suffisant sur toute la base, i.e. sur $D_t = \bigcup D_t^u ... D_t^v$. Les résultats sont récupérés par le pair $Distributed_{SP}$ (i.e. fonction "GetValuation"). L'heuristique, basée sur des opérateurs génétiques est alors appliquée et le nouvel ensemble de candidats est envoyé aux pairs connectés pour évaluation. Ce processus est répété tant qu'il existe des nœuds connectés.

$Distributed_{SP}$ débute lorsqu'un nœud u_t se connecte ($(recvu_t)$). L'ensemble des motifs fréquent est alors initialisé avec la séquence de u_t. Tant que des nœuds sont disponibles, nous considérons les motifs envoyés à $Distributed_{SP}$ par la fonction $getValuation$ afin de déterminer si une séquence est fréquente. *SCORE* correspond à une note moyenne donnée par tous les nœuds pour les candidats. Si *SCORE* est plus grand ou égal à la valeur de support, le candidat devient fréquent et est stocké dans F_{D_t}. Nous conservons également les sous séquences candidates non fréquentes, appelées séquences approximatives et stockées dans $\tilde{F_{D_t}}$,

Algorithme 1: Algorithme $Distributed_{SP}$

Data : F_{D_t} Les séquences fréquentes pour un nœud u_t, les nœuds u_t ...v_t connectés et un paramètre $mindist$ correspondant à la distance minimale d'un candidat pour être considéré.
Result : F_{D_t} les motifs fréquents correspondant aux comportements fréquents sur D_t.

```
recv (u_t);
F_{D_t} ← D_t^u;
while des nœuds sont connectés do
    nodesAvailable ← recv(u_t...v_t);
    F̃_{D_t} ← ∅;
    candidates ← getValuation(nodesAvailable);
    for c ∈ candidates do
        if SCORE(c) > minsupp then
            insert(c, F_{D_t});
        else
            if SCORE(c)) ≥ mindist then
                insert(c, F̃_{D_t});
    candidates ← neighborhoodOperators(F̃_{D_t}, F_{D_t});
    Broadcast(@nodesAvailable, candidates);
```

dont la taille par rapport à la taille totale des séquences candidates vérifie une contrainte de distance spécifiée par l'utilisateur $mindist$. Ces séquences seront utilisées pour les phases de génération de candidats. Grâce aux séquences fréquentes approximatives et aux opérateurs de voisinage, de nouveaux candidats sont générés et envoyés par broadcast aux nœuds connectés. Par manque de place, nous ne décrivons pas les opérateurs de voisinage utilisés. Le lecteur intéressé peut se reporter à Masseglia et al. (2003) où nous utilisons, dans un autre contexte, des opérateurs génétiques similaires.

Deux opérations principales sont réalisées dans l'algorithme $node$. Premièrement, lorsqu'un nouveau pair v_t essaye de se connecter à un nœud u_t (recv(v,$connect$)), un message lui indiquant l'adresse de $Distributed_{SP}$ lui est retourné. Deuxièmement, lorsqu'un message de $Distributed_{SP}$ est reçu, un score représentant la distance entre un candidat et les opérations locales effectuées sur le nœud est calculé. Si un candidat est inclus, son score est positionné à $100 + size(candidate)$. Comme notre approche est heuristique, nous récompensons fortement les candidats complètement inclus dans une séquence. En outre comme nous recherchons les comportements les plus longs, nous récompensons les longues séquences. Ceci est réalisé par l'algorithme Longest-Common-Subsequence (LCS) Cormen et al. (2001).

4 Experimentations

Pour évaluer notre approche, différentes expériences ont été menées sur des jeux de données réelles : le fichier de données "Pumsb" Repository et un fichier d'access log (AccessLog). Les premières expériences ont été réalisées pour analyser la convergence des résultats ainsi que les coûts de communication. Dans le premier cas, un algorithme traditionnel de recherche de motifs a d'abord été appliqué sur la base globale. Chaque population de candidat proposée par notre approche est alors comparée au résultat réel de manière à déterminer sa qualité. Pour cela, nous mesurons pour chaque population de candidat, la plus longue sous séquence

Algorithme 2: Algorithme Node

Data : CS la séquence candidate à évaluer et D_u^t la partition du nœud u au temps t.
Result : $LSCORE$ l'ensemble des scores locaux attribués à chaque séquence.
if *(recv(v,connect)* **then**
  send(v,@$Distributed_{SP}$);
if *(recv(*@$Distributed_{SP}$,CS*)* **then**
  for $S \in CS$ **do**
    $LSCORE$[S] $\leftarrow \emptyset$;
    if *(*$S \subseteq D_u^t$*)* **then**
      $LSCORE[S] \leftarrow$ 100+size(S);
    else
      //Donner à S une note et favoriser les longues séquences.
      $LSCORE[S] \leftarrow \frac{size(LCS(S,D_u^t))*100}{size(S)} - size(S)$;
  send(@$Distributed_{SP}$, $LSCORE$);

commune (LCS) entre les séquences candidates et le résultat réel. Puis la base est partitionnée en différents nœuds et notre approche est appliquée. Les résultats de l'expérience sont décrits par la figure 2. Nous remarquons que pour Pumsb, à la première génération, la qualité de la population candidate est supérieure à 50%. A la seconde, nous avons 70%. Pour les deux jeux de données, à partir de la génération 6, la qualité du résultat est proche de 95% et nous devons attendre la génération 7 pour avoir 100%. Ces résultats montrent que notre approche peut rapidement obtenir des longues séquences fréquentes en ne réalisant que 7 opérations de broadcast.

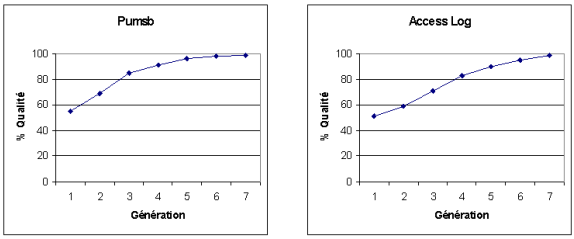

FIG. 2 – *Qualité des résultats pour les populations proposées*

Pour évaluer le comportement de notre approche lorsque des nœuds apparaissent ou disparaissent, nous avons considéré les deux jeux de données. L'idée principale est la suivante : nous voulons analyser le comportement de notre approche lorsque x% séquence de la base d'origine Pumsb sont remplacés par x% séquences de la base destination AccessLog. Nos expériences consistent donc à estimer la qualité des résultats par rapport à un algorithme traditionnel une fois que toute la base a été modifiée. Pour simuler un comportement réaliste du système, nous avons procédé à des remplacements par intervalle de 1-3% à chaque génération. Les résultats ont montré que lorsque Pumsb était remplacé à un rythme de 1% par génération, la qualité des

résultats à la fin du processus de remplacement est de 100%.

5 Conclusion

Dans cet article, nous proposons une nouvelle approche pour améliorer la localisation de ressources dans des systèmes P2P non structurés. Cette approche est inspirée des algorithmes génétiques pour retrouver efficacement les séquences fréquentes dans les nœuds du réseau. Les expériences réalisées ont montré que cette approche est efficace d'une part pour retrouver les comportements fréquents (100% des fréquents sont déterminés en 7 générations quelque soit leur longueur) mais également pour prendre en compte les évolutions dans le réseau (modification forte du comportement des nœuds).

Références

Cormen, T., C. Leiserson, R. Rivest, et C. Stein (2001). *Introduction to Algorithms*. MIT Press.

Masseglia, F., M. Teisseire, et P. Poncelet (2003). HDM : A client/server/engine architecture for real time web usage mining. *Knowledge and Information Systems 5*(4), 439–465.

Miller, M. (2001). *Discovering P2P*. Sybex Inc.

Ng, T., Y. Chu, S. Rao, K. Sripanidkulchai, et H. Zhang (2003). Measurement-based optimization techniques for bandwitdth-demanding peer-to-peer systems. In *Proceedings of the IEEE International Conference INFOCOM*.

Ngan, T.-W. J., D. S. Wallach, et P. Druschel (2003). Enforcing fair sharing of peer-to-peer resources. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS 03)*, Berkeley, California.

Pei, J., J. Han, H. Pinto, Q. Chen, U. Dayal, et M. Hsu (2001). Prefixspan : mining sequential patterns efficiently by prefix projected pattern growth. In *Proceedings of the International Conference on Data Engineering (ICDE 01)*, Heidelberg.

Repository, F. Workshop on frequent itemset mining implementations (FIMI 04). http ://fimi.cs.helsinski.fi/fimi04.

Srikant, R. A. R. (1995). Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE 95)*, Tapei, Taiwan.

Summary

With the huge number of information sources available on the Internet, Peer-to-Peer (P2P) systems offer a novel kind of system architecture providing the large-scale community with applications for file sharing, distributed file systems, distributed computing, messaging and real-time communication. In this paper we propose a new approach for improving resource searching in an unstructured P2P system. By using a genetic-inspired algorithm, we propose to extract patterns occurring in a large number of nodes. Such a knowledge is very useful for proposing the user with often downloaded or requested files according to a majority of behaviors. It may also be useful in order to avoid extra bandwidth consumption.