

Gestion de données dans les systèmes Pair-à-Pair

Bruno DEFUDE

Dept Informatique

GET- Institut National des Télécommunications

<http://www-inf.int-evry.fr/~defude/P2P>

Plan

1. Introduction
2. P2P pur : GNUTELLA
3. Super-peers
 1. SuperPeers
 2. Kazaa
4. P2P « structuré »
 1. Chord
 2. Autres (P-Grid, CAN, Pastry)
5. P2P sémantique
 1. Routing Indices
6. P2P et BD
 1. Médiation (Piazza)
 2. Évaluation de requêtes (PIER)
7. Synthèse

Moteurs de recherche

- gestion centralisée (index) des ressources
- Index basé sur le contenu des ressources (texte), nom des ressources (images, vidéo), méta-données?
- indexation (robot se baladant sur le web et collectant les données) : index incomplet et pas à jour
- Recherche : algorithme de comparaison requête – document : Vector Space Model (le plus simple), PageRank (Google, tient compte de l'analyse des liens entre documents)

Moteurs de recherche (bilan)

- Difficulté à passer à l'échelle
 - Google est incomplet et nécessite plus de 10000 serveurs pour tenir la charge : difficile à mettre en place
- Pas forcément adapté pour autre chose que du texte
- Centralisé donc sensible aux attaques et aux fautes


Définition P2P

- Chaque nœud participant peut être client et serveur
- Chaque nœud paye sa participation en donnant accès à une partie de ses ressources
- Propriétés :
 - Pas de coordination centralisée
 - Pas de BD centralisée
 - Aucun nœud n'a une vision globale du système
 - Comportement global émerge à partir des interactions locales
 - Tous les services et données sont accessibles de n'importe quel nœud
 - Nœuds sont autonomes
 - Nœuds et connections sont non fiables

Classes de systèmes P2P

- P2P Hybrides (e.g Napster)
 - Index centralisé (non tolérant aux fautes)
 - Échange d'information direct
- P2P « Purs » (e.g Freenet, Gnutella)
- P2P Hiérarchiques ou « super-peers » (e.g Kazaa)
 - Mélange C/S et P2P
- P2P sémantiques (e.g Routing Indices)
 - P2P « pur » avec routage basé sur une information sémantique

Fonctionnalités d'un système P2P

- Découverte de ressources
- Gestion des mises à jour 
- Passage à l'échelle
- Tolérance aux fautes
- sécurité

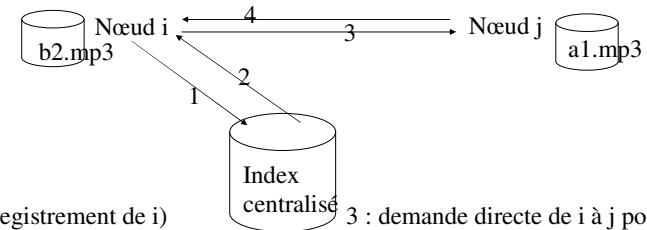
Classes d'applications

- Partage de fichiers : Napster, Gnutella, Freenet, KaZaa
- Système de stockage persistant à grande échelle : OceanStore
- Grid computing : Seti@home

Critères de comparaison

- Recherche de ressource :
 - Topologie du réseau : ouverte (Gnutella) ou contrôlée (Chord)
 - Placement des données et méta-données : libre (Gnutella) ou dirigé (Chord)
 - Routage des messages : fonction de choix des successeurs
- Besoins applicatifs :
 - Expressivité du « langage de requêtes » : égalité (Chord), préfixe (P-Grid), SQL (?), ...
 - Complétude des résultats : une réponse, toutes les réponses, les k meilleures réponses, ...
 - Autonomie des nœuds : choix des ressources à stocker, choix des nœuds successeurs, ...

Napster



- 1: (enregistrement de i) recherche de a1.mp3
 2: retour des nœuds possédant a1.mp3
 3: demande directe de i à j pour télécharger a1.mp3
 4: téléchargement de a1.mp3 et ajout dans la base de ressources partagées

Bilan Napster

- Fonctionne bien à l'échelle d'Internet
- Accès en P2P mais recherche centralisée
- Serveur doit être bien dimensionné et tolérant aux fautes (cluster avec 100, 1000, ... nœuds)
- Sensible aux partitionnements du réseau (serveur inatteignable) et aux attaques

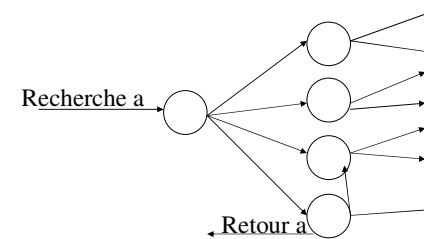
Plan

1. P2P pur : GNUTELLA
2. Super-peers
3. P2P « structuré »
4. P2P sémantique
5. P2P et BD
6. Synthèse

P2P « pur »

GNUTELLA

Gnutella



Chaque nœud propage la requête à k voisins (4)
 Nombre de propagation limité (7)
 Détection de cycles

Gnutella : types de messages

Type	Description	Information
Ping	Annonce disponibilité et lance recherche nouveaux pairs	vide
Pong	Réponse à un ping	Adresse IP + No port; nombre et taille de fichiers partagés
Query	Requête	Bande passante minimum demandée; critère de recherche
QueryHit	Réponse à Query si on possède la ressource	Adresse IP + No port et bande passante; nombre de réponses + descripteurs réponses
Push	Demande de téléchargement pour pairs derrière un firewall	Identifiant du pair; index du fichier demandé; adresse IP No port où envoyer le fichier

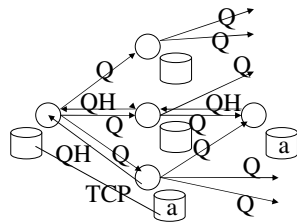
Gnutella (ajout d'un nœud)



Initialisation de la table des nœuds connus (extérieur au protocole)

Compléter la table des nœuds connus

Gnutella (recherche)



BDA04 Montpellier

17

Les principes sous-jacents

- Principe d'égalité entre les nœuds
 - Même capacité (puissance, bande passante, ...)
 - Même comportement (également client et serveur) et bon comportement (pas de « mensonge »)
- Principe de requêtes « populaires »
 - Requêtes concernent principalement peu de ressources
 - Ressources très demandées sont très répliquées
- Principe de topologie du réseau
 - Graphe minimisant le nombre de chemins entre deux nœuds
 - Longueur du chemin minimum entre deux nœuds quelconque est faible (5 à 8)

BDA04 Montpellier

18

Quid des principes en réalité?

- Principe d'égalité entre les nœuds
 - A- [Saroui et al. 01] montrent un écart de 1 à 3 dans la bande passante disponible
 - B- [Adar, Huberman 00] montrent que 70% des utilisateurs ne partagent aucun fichier (ils n'en ont pas ou bien ils n'intéressent personne) et que 50% des résultats sont produits par 1% des nœuds
 - A peut perturber le réseau et produire des partitionnements (trop forte charge demandée à des nœuds connectés via des modems e.g)
 - B implique que d'une part ceux qui partagent n'y ont pas intérêt (pas de réciprocité) et d'autre part que le réseau est sensible aux pannes et aux attaques
 - Études montrent également que certains nœuds sous-évaluent leur bande passante disponible pour éviter d'être choisis

BDA04 Montpellier

19

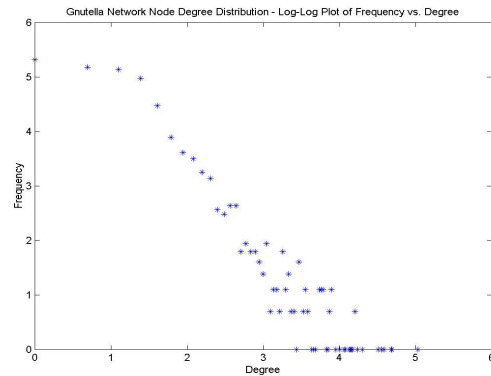
Quid des principes (2)

- Principe des requêtes « populaires »
 - Les 100 requêtes les plus fréquentes sont distribuées uniformément
 - Les autres suivent une distribution de type Zipf
 - Les techniques de cache de résultats s'appliquent bien et peuvent apporter une amélioration notable
- Principe de topologie du réseau
 - Plusieurs études montrent que le graphe sous-jacent de Gnutella est de type « small-world » et que le degré des nœuds suit une distribution « power law »
 - Le principe de diffusion de Gnutella ne s'adapte pas à SW (beaucoup de messages redondants)

BDA04 Montpellier

20

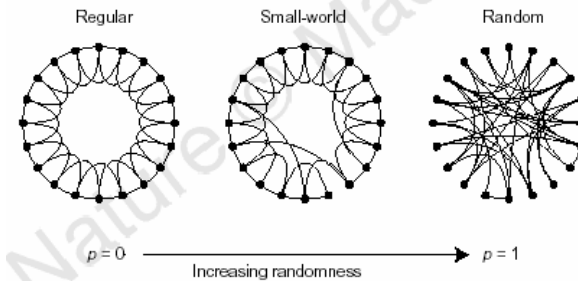
Topologie Réseau gnutella



BDA04 Montpellier

21

Graphes Small World (Watts, Strogatz, Nature)



BDA04 Montpellier

22

Observations sur la bande passante Gnutella

- Sur une période de un mois
 - Requête = 560 bits (y compris headers TCP/IP)
 - Requêtes 25% du trafic, pings 50% et reste 25%
 - En moyenne un pair est connecté activement à 3 autres
- Limite de la dégradation à 10 requêtes / seconde
 - 10 req. X 560 bits X 4 X 3 connections = 67200 b/s
 - Au-dessus de la capacité des modems
- Gnutella ne passe pas l'échelle

BDA04 Montpellier

23

Bilan de Gnutella

- Complètement décentralisé
- Très tolérant aux fautes
- S'adapte bien à la dynamique du réseau
- Simple, robuste et passe l'échelle (pour le moment)
- Gros consommateur de bande passante
- Pas de garantie de succès, ni d'estimation de la durée des requêtes
- Pas de sécurité, ni de réputation
- Problème du « free riding »

BDA04 Montpellier

24

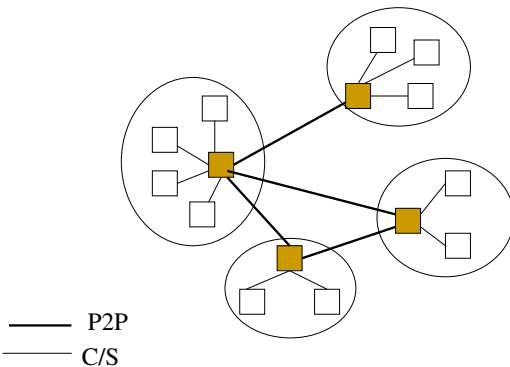
Super-peers

Client/serveur + P2P

Super-peers

- Éviter les problèmes dus à l'hétérogénéité de la bande passante des nœuds
- Tous les nœuds ne sont plus égaux
 - Nœuds avec bonne bande passante sont organisés en P2P : les super-peers
 - Nœuds avec faible bande passante sont rattachés en mode client/serveur à un super-peer (cluster)
 - Super-peers disposent d'un index des ressources de leur cluster
- Utilisé dans KaZaa

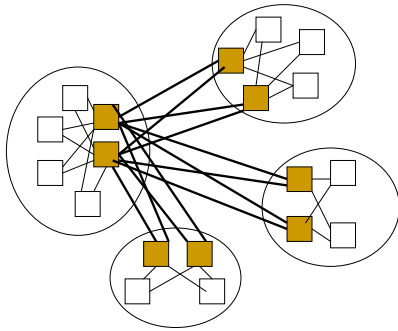
Exemple de super-peer



Super-peer redondant

- Super-peer introduit de la sensibilité aux fautes
- Amélioration possible, choisir k super-peers (partenaires) au lieu d'un dans un cluster
- Chaque partenaire est connecté à chaque client et possède un index de toutes leurs ressources
- Clients envoient leurs requêtes aux partenaires selon un principe de « round-robin »
- Les voisins d'un partenaire distribuent également leurs requêtes équitablement
- Fait baisser la charge du partenaire d'un facteur k
- Augmente le coût d'entrée d'un nouveau client d'un facteur k
- Augmente le nombre de connexions ouvertes de k^2

Exemple de super-peer redondant



BDA04 Montpellier

29

Règles de conception d'un super-peer [Yang 2003]

- R1 : augmenter la taille d'un cluster augmente la charge individuelle (bande passante) mais diminue la charge agrégée
 - Peu de gros clusters : sensibilité aux fautes, attaques, ...
- R2 : la redondance de super-peer est favorable
 - Amène la bonne charge agrégée des gros clusters avec une faible charge individuelle et une bonne tolérance aux fautes

BDA04 Montpellier

30

Règles de conception d'un super-peer

- R3 : maximiser le nombre de connexions des super-peers (diminue le nombre de sauts pour atteindre les résultats). Stratégie doit être choisie par tous les super-peers
- R4 : minimiser le TTL (Time To Live)

BDA04 Montpellier

31

P2P « Structuré » (DHT)

Distributed Hash Table :
Chord, P-Grid

BDA04 Montpellier

32

Chord [Stoica et al. 2001]

- Table de hachage distribuée
- Les nœuds sont répartis sur un anneau
- Les ressources sont réparties sur les différents nœuds de l'anneau
- Structure dynamique
 - Ajout/retrait de nœud
 - Panne d'un nœud
- Peut être utilisée pour construire des applications au-dessus (DNS, ...)

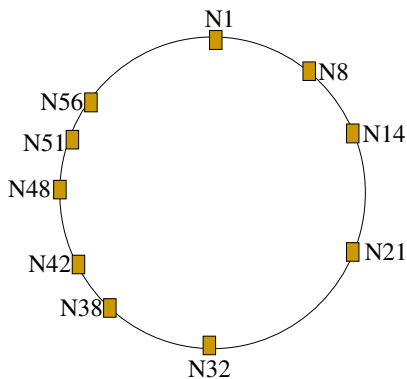
Consistent Hashing

- Chord alloue les ressources aux nœuds en utilisant une fonction de hachage
- Consistent Hashing garantit avec une probabilité élevée
 - Ressources sont distribuées uniformément sur l'ensemble des nœuds
 - L'ajout/retrait d'un Nème nœud n'oblige à déplacer que $O(1/N)$ ressources

Structure en anneau

Chaque nœud est alloué sur l'anneau en fonction de hash(IP)

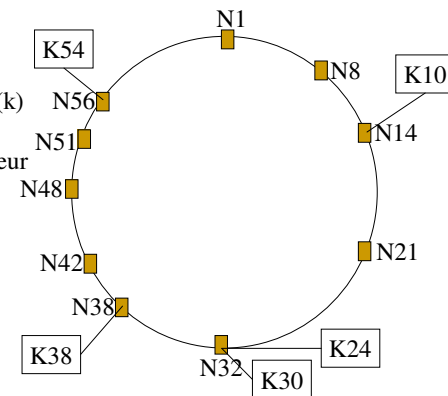
Au plus 2^m nœuds



Placement des ressources

Hash(ressource)=k

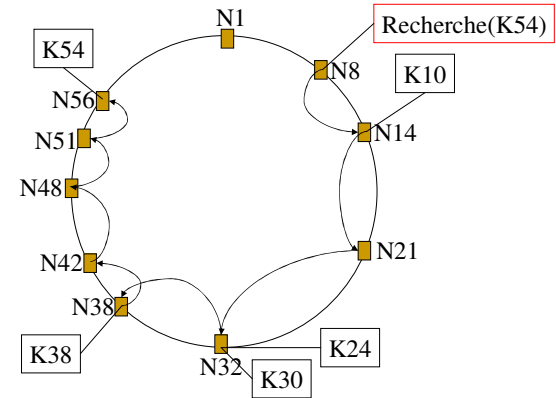
k placé sur successeur(k)
successeur(k)=nœud immédiatement supérieur (ou égal) à k



Recherche (naïve) d'une ressource

- Sur le nœud i on reçoit la requête : recherche(k)
- Si i possède k, il retourne k
- Sinon, il propage la requête à son successeur (chaque nœud doit stocker l'identification de son successeur)
- Le résultat suit le chemin dans l'ordre inverse
- Recherche linéaire en nombre de nœuds

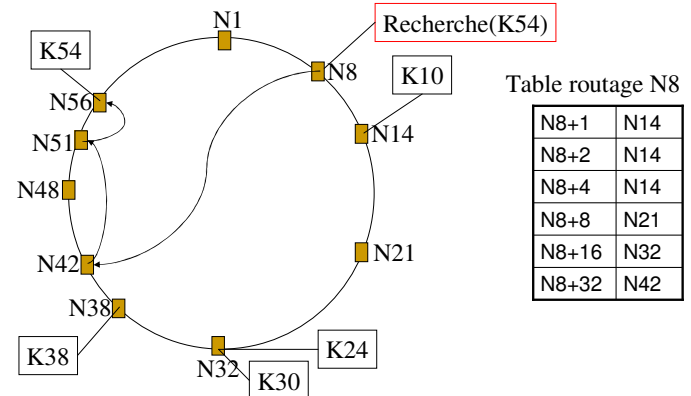
Exemple de recherche naïve



Amélioration de la recherche

- Avoir une table de routage plus complète
- Pour chaque nœud i :
 - Succ[k]=premier nœud sur l'anneau qui vérifie $(i + 2^{k-1}) \bmod 2^m, 1 \leq k \leq m$
 - Successeur=succ[1]
 - m entrées dans la table
 - Prédecesseur (utilisé pour la maintenance dynamique du réseau)
 - Donne adresse IP et No de port du nœud

Exemple de recherche



Algorithme de recherche

- Rechercher si la clé existe localement. Si oui on renvoie la valeur associée sinon
- Rechercher dans table routage nœud avec plus grande valeur inférieure ou égale à la clé cherchée
- Transmettre la requête au nœud sélectionné et appliquer récursivement
- Nombre de sauts moyen : $O(\log_2(N))$

BDA04 Montpellier

41

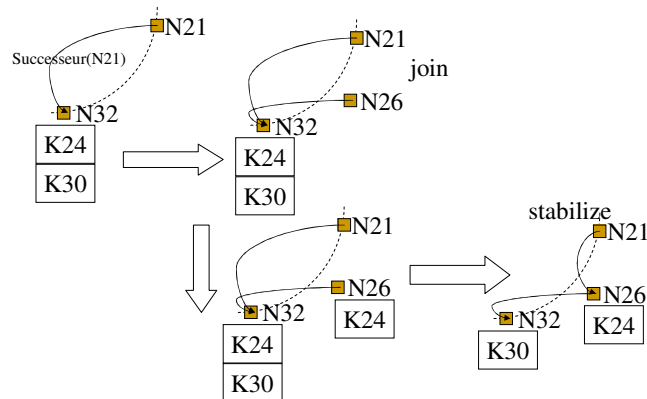
Ajout d'un nœud

- Repose sur la combinaison d'opérations élémentaires
 - N.join(n') : nœud n annonce au nœud n' qu'il rentre dans le réseau et lui demande de lui fournir son successeur
 - N.stabilize() : lancé périodiquement, permet à n et à son successeur de vérifier qu'ils forment un couple correct (il n'y a pas de nouveaux nœuds entre les 2)
 - N.majentrées() : lancé périodiquement, permet d'initialiser la table de routage pour les nouveaux nœuds ou de la mettre à jour pour les nœuds existants
 - N.testpredecesseur() : lancé périodiquement, vérifie que le prédecesseur est toujours là

BDA04 Montpellier

42

Exemple d'ajout N26



BDA04 Montpellier

43

Propriétés de l'algorithme d'ajout

- L'algorithme garantit que n'importe quelle séquence de join entrelacée avec des stabilize converge vers un état stable (tous les nœuds restent atteignables)
- Même en présence d'un ajout (limité) de nœud, le coût de la recherche reste en $O(\log(N))$
- Une recherche peut échouer si le réseau n'est pas complètement stabilisé (il faut relancer la recherche un peu plus tard)

BDA04 Montpellier

44

Sensibilité aux fautes

- Algorithme de recherche repose sur la notion de successeur (sensible à la panne de celui-ci)
- Une solution consiste à gérer une liste de r successeurs

Evaluation Chord

- Type de recherche : égalité
- Coût de la recherche : $O(\log(n))$
- Coût de la mise à jour : $O(\log(n))$
- Coût de l'ajout d'un nœud : $O(\log_2(n))$
- Pas d'autonomie de stockage et de routage

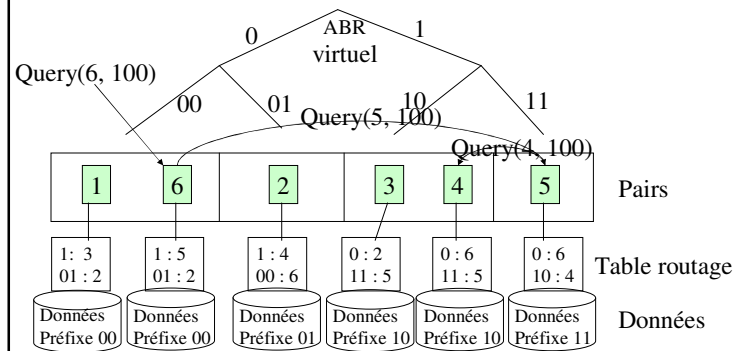
Bilan Chord

- Algorithme assez simple, avec de bonnes propriétés démontrables
- Résultats expérimentaux confirment
- Problème de latence :
 - Fonction de recherche minimise le nombre de sauts, mais tous les sauts n'ont pas forcément le même prix (traversée transatlantique e.g)
 - Besoin d'utiliser de l'information sur la distance entre les nœuds (on choisira parmi les successeurs possibles celui à distance minimale) -> Global Network Positioning

P-Grid [Aberer 01]

- Table de hachage distribuée
- Nœuds se répartissent les ressources selon une structure d'arbre binaire de recherche
- Repose sur une fonction de hachage préservant les préfixes et distribuant uniformément les ressources sur l'arbre binaire
- Structure auto-adaptative, avec réplication
- Permet les recherches basées sur les préfixes (pas seulement égalité stricte)

Exemple d'un P-Grid



BDA04 Montpellier

49

Bilan de P-Grid

- Structure assez simple, tolérante aux fautes, avec passage à l'échelle
- Évite de structurer les nœuds selon une fonction de hachage
- Peut intégrer facilement des optimisations sur la latence (choix d'un pair)
- N'est pas restreint aux recherches basées sur l'égalité stricte

BDA04 Montpellier

50

Autres approches de DHT

- Freenet [Clarke 01]
- CAN [Ratsanamy 01] : espace à n-dimensions
- Pastry [Rowstron 01] : hypercube

BDA04 Montpellier

51

P2P « sémantique »

Routing Indices

BDA04 Montpellier

52

P2P « sémantique »

- Ajouter de l'information aux tables de routage
- Généraliste vs spécifique (à une application)
- Information doit pouvoir être maintenue dynamiquement
- Équilibre entre taille et précision

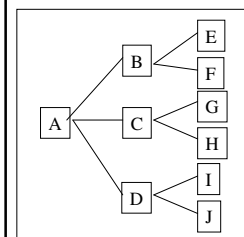
Quelle information ?

- Sur le contenu des nœuds (index)
 - Suppose que les nœuds aient un contenu homogène (pas de placement aléatoire)
 - Routage par comparaison requête-index
 - Même information partagée par tous, ou bien information construite localement
 - Exemple « routing indices »
- Sur le réseau (clustérisation)
 - Regrouper logiquement les nœuds avec même « sémantique »
- Sur les requêtes
 - Suppose qu'il y ait peu de requêtes différentes
 - Routage par comparaison requête-requête
- Sur les utilisateurs
 - Suppose que les utilisateurs aient toujours le même besoin
 - Routage par comparaison utilisateur-utilisateur

« Routing Indices » [Crespo 02]

- Introduire de l'information sur le contenu des nœuds (index)
- Analogue aux systèmes d'index répartis et hiérarchisés pour moteurs de recherche sur Internet
- Trouver l'équilibre entre la taille de l'index et le gain

Exemple de « RI »



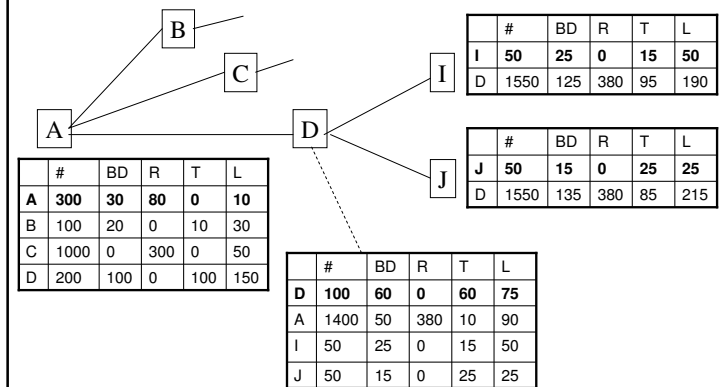
RI pour nœud A

Chemin	Nb documents	BD	Réseaux	Théorie	langages
B	100	20	0	10	30
C	1000	0	300	0	50
D	200	10	0	100	150

Utilisation de l'index

- Soit Q une requête, conjonction de plusieurs termes de recherche (t_Q^1, \dots, t_Q^k)
- Proximité(Q, chemin) = $Nbdedocuments \times \prod_i (RI(t_Q^i)/Nbdedocuments)$
- Q émise sur A = ('BD', 'langages')
- Proximité(Q, B) = $100 \times 20/100 \times 30/100 = 6$
- Proximité(Q, C) = $1000 \times 0/1000 \times 50/1000 = 0$
- Proximité(Q, D) = $200 \times 100/200 \times 150/200 = 75$
- Permet d'ordonner les nœuds successeurs

Exemple complet de RI



Algorithme de recherche

- Résoudre Q localement. Si suffisamment de résultats OK, sinon
- Tant qu'il n'y a pas assez de résultats
 - Évaluer proximité des successeurs
 - Prendre le successeur S le plus proche (et non encore exploré), si vide retour
 - Recherche(Q, S)

Performances de la recherche

- Par rapport à Gnutella diminue le nombre de messages
- Exploration restreinte aux nœuds ayant la plus grande probabilité de succès
- Pas d'information sur le nombre de sauts nécessaires (améliorations possibles avec d'autres RI)
- Pas de garantie d'avoir tous les résultats
- Plutôt orienté recherche des k meilleurs résultats

Création d'un RI (ajout de AD)

Diagram illustrating the creation of an RI (ajout de AD). The graph shows nodes A, B, C, D, I, and J. Node A is updated with values from nodes B, C, and D. Node D is updated with values from nodes I and J. The resulting RI table for A is shown with values 1400, 50, 380, 10, 90.

#	BD	R	T	L	
A	300	30	80	0	10
B	100	20	0	10	30
C	1000	0	300	0	50
D	100	60	0	60	75
I	50	25	0	15	50
J	50	15	0	25	25

1400, 50, 380, 10, 90

200, 100, 0, 100, 150

BDA04 Montpellier 61

Création d'un RI (2)

Diagram illustrating the creation of an RI (2). The graph shows nodes A, B, C, D, I, and J. Node A is updated with values from nodes B, C, and D. Node D is updated with values from nodes I and J. The resulting RI table for A is shown with values 1550, 125, 380, 95, 190.

#	BD	R	T	L	
A	300	30	80	0	10
B	100	20	0	10	30
C	1000	0	300	0	50
D	200	100	0	100	150
I	50	25	0	15	50
J	50	15	0	25	25

1550, 125, 380, 95, 190

1550, 135, 380, 85, 215

Somme de D sauf J

A Maj sur B et C

BDA04 Montpellier 62

Maj d'un RI

- Analogue au processus de création
- Pour diminuer le coût, on peut accumuler les maj et les traiter par lots
- Suppression d'un nœud suit le même principe

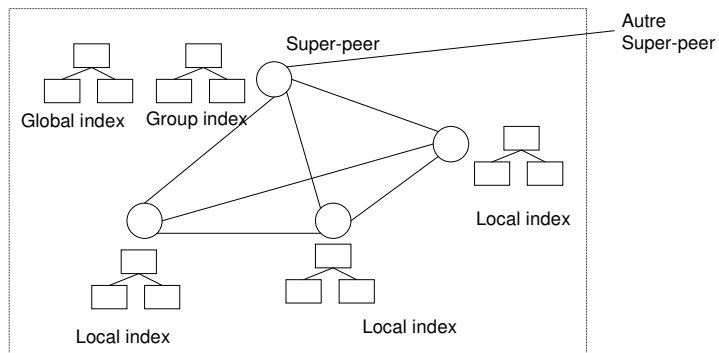
BDA04 Montpellier 63

Bilan RI

- Structure d'indexation assez simple, mais maj génère beaucoup de messages
- Fonctionne bien pour obtenir les **meilleurs** résultats, mais pas forcément **tous** les résultats
- Pour traiter des graphes généraux, il faut intégrer la gestion des cycles (détection ou prévention)
- S'applique à des langages types mots-clés mais pas généralisable à des langages plus complexes
- RI ne donne pas d'indications sur le nombre de sauts (d'autres RI sont proposées, e.g distinguer le nombre de sauts dans la RI)
- Classifier l'ensemble des nœuds via une classification « sémantique » (e.g genres de musique)

BDA04 Montpellier 64

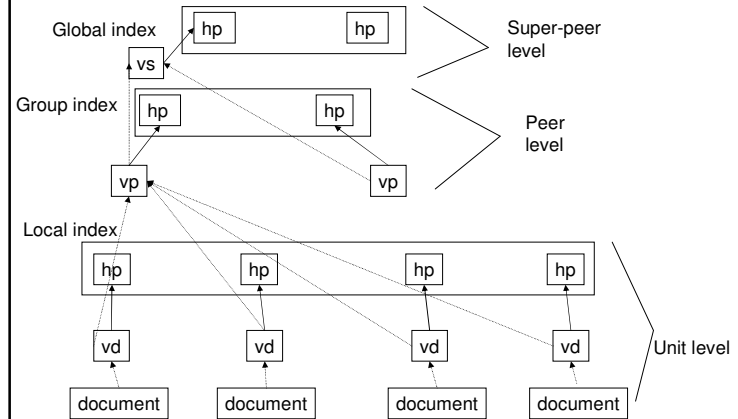
Une approche plus générale [Shen 04]



BDA04 Montpellier

65

Hierarchie de résumés



BDA04 Montpellier

66

Construction des résumés

- Construction en deux étapes
 - VSM : chaque document est représenté par un vecteur (de grande dimension)
 - LSI : utilisation de SVD, donne un point dans un espace à k dimensions
- index
 - Adaptation des VA files (Vector Approximation)
 - Permet de calculer efficacement les K plus proches voisins

BDA04 Montpellier

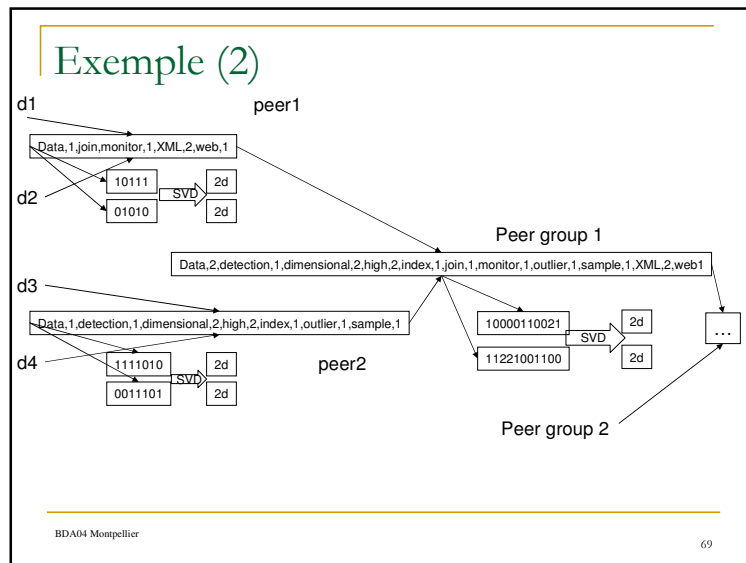
67

Exemple (1)

	Id	Documents
P1	d1 ₁ ¹	Monitoring XML Data on the Web
	d2 ₁ ¹	Approximate XML joins
P2	d3 ₂ ¹	Outlier Detection for High Dimensional Data
	d4 ₂ ¹	High Dimensional Indexing using Sampling
P3	d5 ₁ ²	Document clustering with committes
	d6 ₁ ²	Document clustering with cluster refinement
P4	d7 ₂ ²	Title language model for information retrieval
	d8 ₂ ²	Document summarization in information retrieval

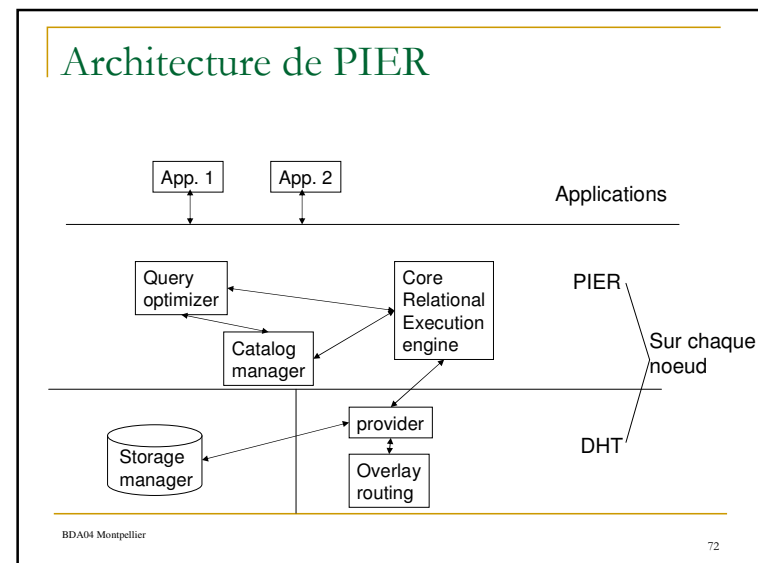
BDA04 Montpellier

68



- ### Peer-ot-Peer et bases de données
- BD dans les systèmes P2P
 - Augmenter la puissance d'expression du langage de requêtes (Huebsch 03)
 - Gestion de caches de résultats de requêtes (Sahin 04)
 - P2P dans les SGBD
 - Évaluation de requêtes réparties (Papadimos02, Nejd1 03)
 - Intégration de schémas (Tatarinov 03, Ooi 03)
 - « intégration » de données (Arenas 03)
- BDA04 Montpellier 70

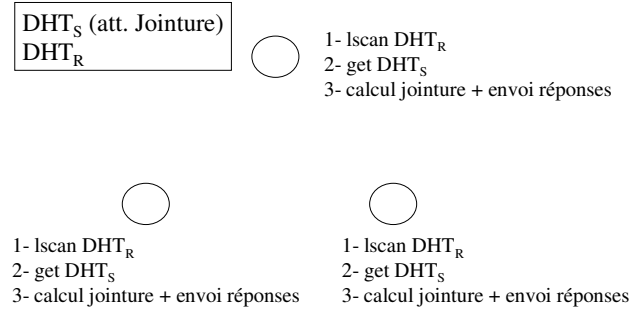
- ### Projet PIER (Berkeley)
- Peer-to-peer Information Exchange and Retrieval
 - Moteur de requêtes pour des environnements distribués à très grande échelle (plusieurs milliers de nœuds)
 - Type d'applications visés : surveillance de l'Internet
- BDA04 Montpellier 71



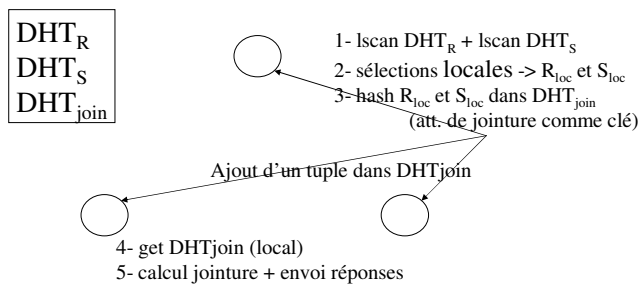
Architecture PIER (2)

- DHT : Bamboo, CAN, Chord
- Routing : leave, join, lookup(key), locationMapChange()
- Storage manager : store, retrieve, remove
- Provider : get, put, renew, multicast, lscan, newData

Algorithmes d'équi-jointure (1)



Algorithmes d'équi-jointure (2)



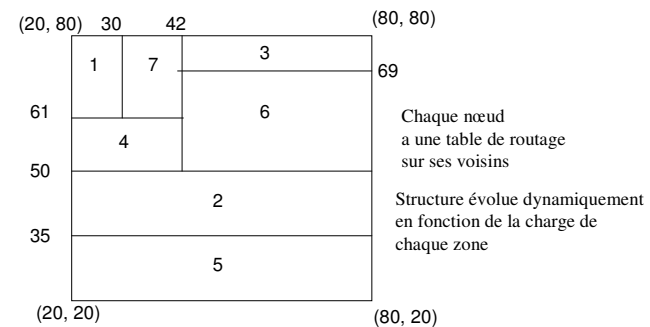
Bilan PIER

- Permet une évaluation de requêtes relationnelles sur des DHT
- Passage à l'échelle
- Intégrer d'autres opérations (range queries, θ -jointure)
- Beaucoup de travail à faire sur l'optimisation

Gestion de cache pour «range queries»

- Gérer un cache de résultats de « range queries » et accéder à ce cache pour résoudre une nouvelle requête
- Utiliser une DHT (CAN) pour stocker et adresser les résultats (indexés par les bornes inf et sup de la recherche)

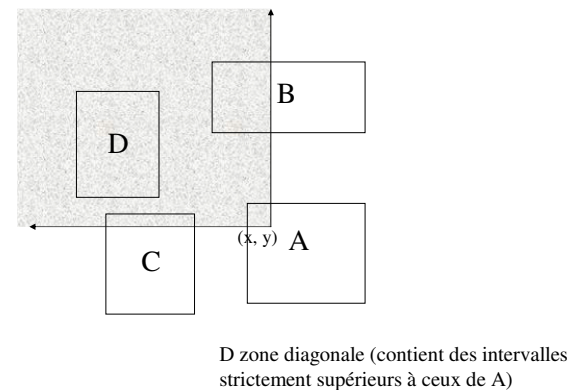
Exemple d'espace de recherche (CAN à 2 dimensions)



Utilisation de la DHT

- $Q(\text{binf}, \text{bsup})$ est émise sur un nœud
- Elle est routée sur la zone correspondante
- Recherche si des résultats contiennent Q (soit stockés localement, soit indexés localement)
- Si vide, propagation de Q

Propagation d'une requête



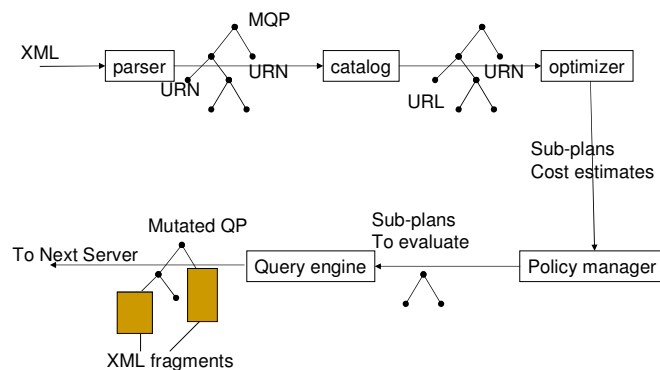
Bilan

- CAN s'applique bien ici (se généralise aussi facilement à des attributs multiples)
- Évaluations montrent un bon comportement
- Encore beaucoup de choses à faire!!

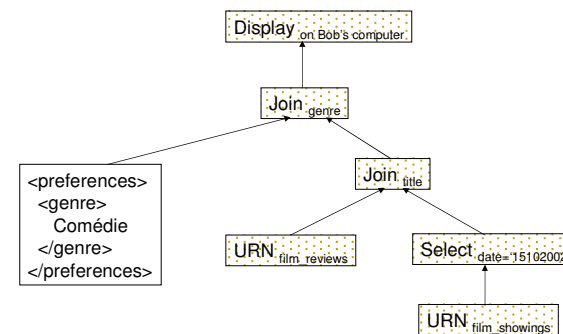
Mutant Query Plans (Papadimos 02)

- Évaluation répartie de requêtes à large échelle
- Principe de « requête mobile » : Mutant Query Plan (MQP) se balade de site en site en construisant petit à petit le résultat
- $MQP_{initial}$ = requête
- MQP_{final} = résultats

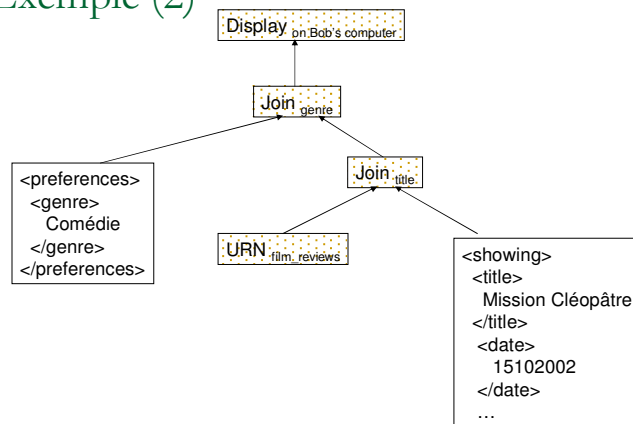
Principe d'évaluation



Exemple (1)



Exemple (2)



Bilan MQP

- Application des idées de code mobile à l'évaluation de requêtes réparties
 - Avantages : passe bien l'échelle, plus simple que le code mobile (un MQP est un programme très spécialisé)
 - Inconvénients : les résultats sont plus gros en BD, plutôt un mode batch
- Les performances dépendent beaucoup de la « qualité » du catalogue (connaissance des sites qui peuvent résoudre, au moins partiellement, la requête) et du choix associé

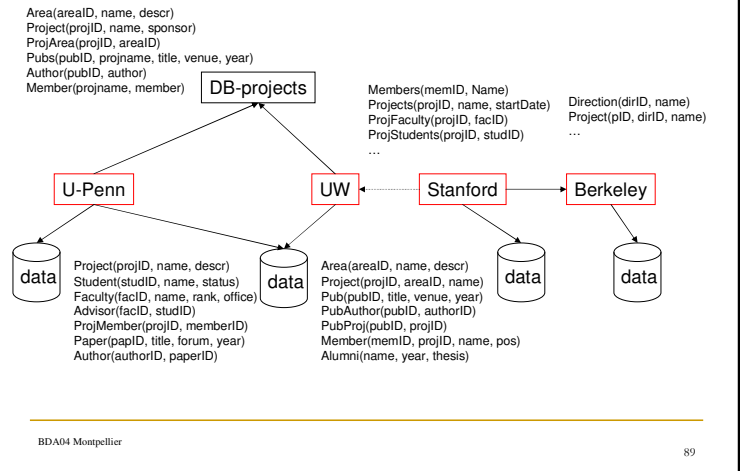
Projet Piazza (Washington Univ.)

- Système de médiation sans schéma global
- Chaque source dispose de son propre schéma et de mappings vers d'autres sources
- composition de mappings
- Résolution de requêtes : problème de reformulation (approche mixte LAV et GAV)

Piazza (problèmes abordés)

- Définition du langage d'expression des mappings
- Étudier la composition de mappings
- Étudier la reformulation de requêtes
- Optimiser le processus d'évaluation (pipeline, pré-calculs)
- Dans un cadre relationnel et XML

Exemple de PDMS Piazza



Définition des mappings (PPL)

- Description des sources (storage description)
 - $A : R = Q$ ou $A : R \subseteq Q$ (A pair, R relation stockée par A, Q requête définie sur le schéma de A)
 - UPenn : students(sid, name, advisor) \subseteq UPenn : Student(sid, name, -), UPenn : Advisor(sid, fid), UPenn : Faculty(fid, advisor, -, -)
- Médiation (peer mappings)
 - $Q_1(A_1) = Q_2(A_2)$ ou $Q_1(A_1) \subseteq Q_2(A_2)$
 - Q_1 et Q_2 requêtes conjonctives de même arité
 - A_1 et A_2 ensemble de pairs
 - Definitional mappings
 - Règles datalog portant sur les relations des pairs

Complexité de l'évaluation de requêtes

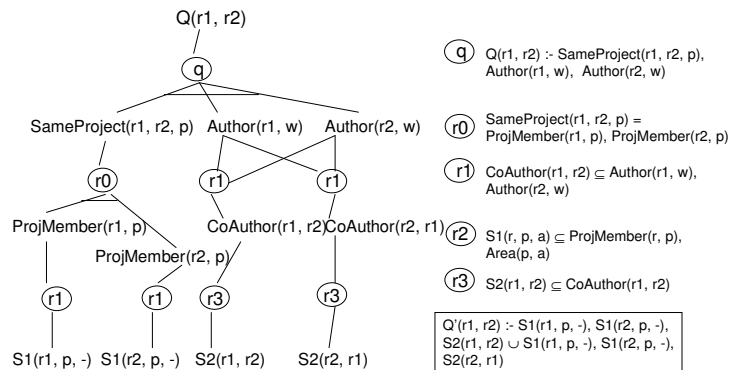
Soit N un PDMS spécifié en PPL

- Trouver toutes les réponses certaines à une requête conjonctive Q pour N est indécidable
- Si N contient uniquement des mappings non cycliques, alors une requête conjonctive peut être évaluée en un temps polynomial

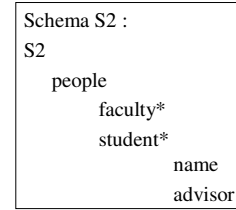
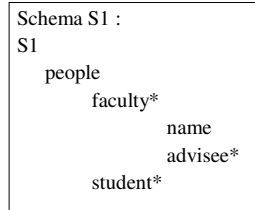
Algorithme d'évaluation de requêtes

- Transformer Q en Q' (exprimée sur des relations stockées uniquement)
- Suppose que les mappings sont connus globalement!!
- Selon les mappings utilisés, doit mixer évaluation GAV et LAV

Exemple de graphe de reformulation (DAG)



Piazza (schémas XML)



Mapping S1 vers S2

```

<S2>
for $people in /S1/people
return
  <people>
    for $name in $people/faculty/name/text()
    return
      <faculty>{$name}</faculty>
    for $student in $people/student/text()
    return
      <student>
        <name>{$student}</name>
        for $faculty in $people/faculty,
        $name in $faculty/name/text(),
        $advisee in $faculty/advisee/text()
        where $advisee=$student
        return
          <advisor>{$name}</advisor>
      </student>
    </people>
</S2>

```

Restriction de XQuery
Sémantique ensembliste

Composition de mappings

- Trois paires A, B, C et M1 mapping de A vers B et M2 mapping de B vers C
- Cas 1 : $M_1 : A \supseteq Q_1(B)$ et $M_2 : B \supseteq Q_2(C)$
 - $M_1 \circ M_2 : A \supseteq Q_1 \circ Q_2(C)$
- Cas 2 : $M_1 : B \subseteq Q_1(A)$ et $M_2 : B \supseteq Q_2(C)$
 - $M_1 \circ M_2 : A \supseteq M_1^{-1} \circ Q_2(C)$
 - $M : A \subseteq Q(B)$ alors $M^{-1} : B \supseteq [I_B]_Q(A)$

Exemple de calcul d'inversion

```

for $S1 in /S1 return
<S1> {
  for $people in $S1/people return
  <people> {
    for $faculty in $people/faculty return
    <faculty> {
      for $name in $faculty/name/text() return
      <name> {$name} </name>
      for $advisee in $faculty/advisee/text() return
      <advisee> {$advisee} </advisee>
    } </faculty>
    for $student in $people/student/text() return
    <student> {$student} </student>
  } </people>
} </S1>

```

Calcul d'inversion (suite)

```

for $S1 in /S2/people return
<S1> {
  <people> {
    for $faculty in $people/faculty return
    <faculty> {
      for $name in $faculty/text() return
      <name> {$name} </name>
    } </faculty>
    for $student in $people/student, $advisor in $student/advisor return
    <faculty> {
      for $name in $advisor/name/text() return
      <name> {$name} </name>
      for $advisee in $student/name/text() return
      <advisee> {$advisee} </advisee>
    } </faculty>
    for $student in $people/student/name/text() return
    <student> {$student} </student>
  } </people>
} </S1>

```

Piazza (bilan)

- Médiation sans schéma global
- Travail formel (mappings, composition mappings, réécriture de requêtes, optimisation, ...)
- Sur des données relationnelles et XML
- Certaines parties ne sont pas encore décentralisées (évaluation de requêtes suppose connaissance globale des mappings)

Tentative de Synthèse

Choix P2P, problèmes ouverts

Besoins applicatifs

	Type recherche	résultats	Garantie	Autonomie stockage	Autonomie routage
Gnutella	filtre	Tous?	Non	Oui	Oui
Chord	égalité	Un	oui	non	non
P-Grid	préfixe	Un	oui	Faible	non
Freenet	égalité	Un	Non	Oui	O/N
Super-pair	filtre	K meilleurs	non	Oui	Oui pour SP, non pour autres
sémantique	filtre	K meilleurs	non	Oui	oui

BDA04 Montpellier

101

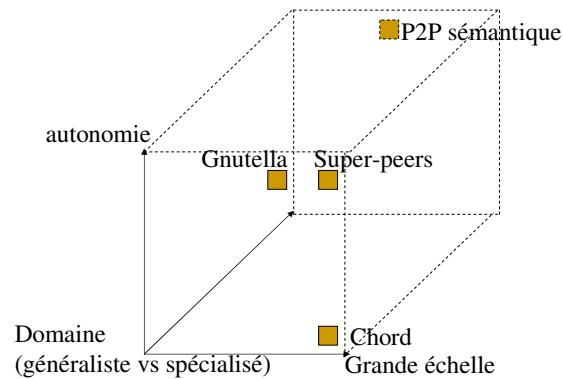
Comparatif performances

	Modèle recherche	Coût recherche (messages)	ajout nœud	MAJ table routage
Gnutella	largeur	$2^* \sum_{i=0}^{TTL} C^*(C-1)^i$	Init. routage	Périodique (ping)
Chord	Arbre binaire recherche	$O(\log(n))$	Maj routage, échange ressources	Périodique (stabilize)
P-Grid	Arbre binaire préfixe	$O(\log(n))$	Maj routage, échange ressources	rencontres
Freenet	profondeur	$O(\log(n)) ?$	Init. routage	Pendant recherche et insertion

BDA04 Montpellier

102

Taxonomie



BDA04 Montpellier

103

Problèmes ouverts

- Sécurité
- réputation
- Contrôler le « free riding »
- P2P sémantique
 - On en est au début
 - Mixer différents types de connaissances
 - Connaissances dynamiques (apprentissage)

BDA04 Montpellier

104

Sécurité

- Disponibilité (résistance aux attaques DoS)
- Authentification des ressources (si plusieurs réponses différentes pour la même ressource)
 - Plus vieille ressource
 - Réputation (« experts »)
 - Vote
- Anonymat
 - Auteur : quels utilisateurs ont créé quels documents?
 - Document : sur quels nœuds est stocké un document donné?
 - Lecteur : quels utilisateurs accèdent quels documents?
 - Serveur : quels documents sont stockés sur un nœud donné?

P2P et BD

- Très actif en ce moment (plusieurs workshops et conférences spécialisés)
- Domaine de recherche multi-disciplinaire (BD, réseau, systèmes répartis, ...)
- Un champ de recherche fructueux pour les BD réparties à grande échelle, ou des modes d'interaction sans intégration de schéma
- Beaucoup de problèmes ouverts

Bibliographie (générale)

- K. Aberer, M. Hauswirth. Peer-to-Peer Information Systems: Concepts and Models, state-of-the-art, and Future Systems, Tutorial IEEE ICDE, 2002
- E. Adar, B. Huberman. Free Riding on Gnutella, Technical Report, Xerox PARC, septembre 2000
- I. Clarke et al. Freenet: A Distributed Anonymous Information Storage and Retrieval System, LNCS 2009, Springer Verlag, 2001
- L. Gong. JXTA: A Network Programming Environment, IEEE Internet Computing, 5(3), mai-juin 2001
- Gnutella : <http://www.gnutella.com>
- M.A. Jovanovic et al Scalability Issues in Large Peer-to-Peer Networks – A Case Study of Gnutella, Research report, Univ. Cincinnati, 2001
- J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective, Technical Report 99-1776, Cornell Univ., 1999
- J. Kubiatowicz et al. OceanStore: An Architecture for Global-Scale Persistent Storage, Proc. ASPLOS, 2000
- S. Saroiu, P. Gummadi, S. Gribble. A Measurement Study of Peer-to-Peer Sharing Systems, Proc. Multimedia Computing and Networking, 2002
- K. Sripanidkulchai. The Popularity of Gnutella Queries and its Implications on Scalability, février 2001
- B. Yang, H. Garcia-Molina. Improving Search in Peer-to-Peer Systems, Proc. 28th Conf. On Distributed Computing Systems, 2002

Bibliographie (DHT et super-pairs)

- K. Aberer et al. Improving Data Access in P2P Systems, IEEE Internet Computing, January 2002
- K. Aberer. P-Grid: A Self-Organizing Access Structure for P2P Information Systems, Proc. COOPIS, 2001
- A. Rowstron, P. Dreschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems, Proc. IFIP/ACM Conf. On Distributed Systems Platforms, 2001
- I. Stoica et al. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications, Proc. ACM SIGCOMM 2001
- S. Ratsanamy et al. A Scalable Content Addressable Network, Proc. ACM SIGCOMM 2001
- B. Yang, H. Garcia-Molina. Comparing Hybrid Peer-to-Peer Systems, Proc. VLDB Conference, 2002
- B. Yang, H. Garcia-Molina. Designing a Super-Peer Network, Proc. ICDE Conf., 2003

Bibliographie (P2P sémantique)

- A. Crespo, H. Garcia-Molina. Routing Indices for Peer-to-Peer Systems, Proc. ICDCS 2002
- P. Haase, R. Siebes. Peer Selection in Peer-to-Peer Networks with Semantic Topologies, Proc. WWW Conf., 2004
- H. T Shen, B. Yu. Efficient Semantic-Based Content Search in P2P Network, IEEE TKDE 16(7), 2004

Bibliographie (médiation P2P)

- K. Aberer, P. Cudré-Mauroux, M. Hauswirth. A Framework for Semantic Gossiping, ACM SIGMOD Record 31(4), 2002
- M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. Miller, J. Mylopoulos. The Hyperion Project: From Data Integration to Data Coordination, ACM SIGMOD Record 32(3), 2003
- A. Halevy, Z. Ives, J. Madhavan, P. Mork, D. Suciu, I. Tatarinov. The Piazza peer data management system. *IEEE TKDE* 16(7), 2004.
- B.C Ooi, Y. Shu, K-L. Tan. Relational Data Sharing in Peer-based Data Management Systems, ACM SIGMOD Record 32(3), 2003
- J. Madhavan, A. Halevy. Composing Mappings Among Data Sources, Proc. VLDB Conf., 2003
- W. Nejdl, W. Siberski, M. Sintek. Design issues and challenges for RDF- and schema-based peer-to-peer systems. *SIGMOD Record*, 32(3), 2003.
- W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmer, T. Rish. EDUTELLA: a P2P Networking Infrastructure based on RDF. *Int. Conf. WWW*, 2002.
- I. Tatarinov, A. Halevy. Efficient Query Reformulation in Peer Data Management System, Proc. ACM SIGMOD 2004

Bibliographie (requêtes complexes et DHT)

- J. Hellerstein. Architectures and Algorithms for Internet-Scale (P2P) Data Management, tutorial VLDB 2004
- R. Huebsch, J. Hellerstein, N. Lanham, B.T Loo, S. Shenker, I. Stoica. Querying the Internet with PIER, Proc. VLDB Conf., 2003
- V. Papadimos, D. Maier. Mutant Query Plans, Information and Software Technology, 44(4), 2002
- O.D Sahin, A. Gupta, D. Agrawal, A. EL Abbadi. A Peer-to-Peer Framework for Caching Range Queries, Proc. ICDE 04 Conf., 2004