

République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université de 08 Mai 1945 – Guelma-
Faculté des Mathématiques, d'Informatique et des Sciences de la matière
Département d'Informatique



Mémoire de Fin d'études Master

Filière : Informatique

Option : Ingénierie des Medias

Thème :

**Gestion des Caches coopératives pour des données
multimédias dans les réseaux mobiles**

Encadré par :

Mme **BENHAMIDA Nadjette**

Présenté par :

KEITA N'faly

KARIANTI Simba

Juin 2013

Remerciement

Au terme de ce travail, nous tenons à exprimer notre gratitude et notre reconnaissance à tous ceux qui ont contribué de près ou de loin à l'élaboration de ce mémoire, en tout premier lieu, notre encadreuse : Mme **BENHAMIDA Nadjette** pour son aide, sa disponibilité, son suivi permanent et ses nombreux conseils qui nous ont permis de mener à bien ce travail.

Nous ne manquerons pas également de montrer notre gratitude à l'ensemble du personnel du département informatique notamment le chef de département Mr. **BERRHOUMA Nabil** et tous les professeurs qui nous ont suivi durant nos études.



Je dédie ce modeste travail.

Aux deux êtres qui me sont très cher, eh oui, ma maman et mon papa, vous qui m'avez élevé et éduqué... vous qui avez toujours été là pour moi et n'avez jamais cessé de croire en moi, aucun mot ni aucune langue ne pourrait exprimer ma profonde gratitude à votre égard.

A toute la famille Simba et Abdou tocha, à mes grands frère kamarzamane et kamardine sans oublier leurs femmes Mariama Abdallah et Chariata Mzé, à mon petit frère Kamal, à mes sœurs Kamaria et Kaissoiri, à mes deux nièces Nawal et Massima, à mon neveu nawad et à la famille de Kenza qui s'est occupé de moi durant mes trois derniers années en Algérie.

A tous mes amis Elia, Oumou koulthoum, Matoya et à tous ceux que j'ai rencontrés en Algérie...

*A tous ceux qui ont veillé à mon instruction
A tous ceux que j'aime et qui m'aiment
Avec l'expression de tous mes sentiments et mon respect,*

Que Dieu accepte notre travail.

Simba Karianti

Dédicaces

Je commence par remercier Dieu le tout puissant et lui rendre grâce.

C'est avec un grand plaisir que je dédie ce modeste travail, fruit de mes études en exprimant ma profonde gratitude à tous mes parents et proches.

Je commence par mon père **KEITA Djiba** qui m'a indiqué la bonne voie en me rappelant toujours que seul le travail paie.

A ma mère **KEITA Kouden**, pour la bonne éducation que j'ai reçue d'elle.

Vous m'avez donné la vie, vous m'avez vu grandir et vous vous êtes sacrifiés pour que je puisse terminer ce deuxième cycle de LMD.

Que ce mémoire soit le fruit de ma reconnaissance et qu'il fasse l'objet de votre fierté.

Je remercie toute ma famille : **Mohamed Kaba Keita, Fanta, Kanko, Nantenin, Djamanaty, Kanko, Camara, mes Beaux-Frères Kaba Kourouma, Sakoba Kourala Keita, Lancinè, Keita** qui n'ont cessé de m'encourager.

Je remercie également mon binôme **KARIANTI Simba** et tous mes collègues et amis de la Guinée, de l'Algérie et dans d'autres pays.

KEITA N'faly

Résumé

Le partage de données dans des systèmes à grande échelle est devenu crucial. De nombreux efforts ont été réalisés et sont menés pour la proposition d'intergiciels efficaces d'interrogation multi-sources largement réparties. Les performances de tels intergiciels dépendent de nombreux facteurs et sont fortement influencées par les caractéristiques de l'infrastructure matérielle.

Ce mémoire présente une solution de caches coopératifs améliorant les coûts d'évaluation de requêtes et de transferts de données dans les systèmes de gestion de données réparties. Cette solution repose sur la décentralisation des données sous une forme de base de données distribuée et donc d'alléger les tâches du serveur et les liaisons entre ce serveur et les autres clients. Les données sont gérées à l'aide de deux caches distincts, stockant séparément les requêtes des objets. Ainsi un cache peut demander la coopération d'un ou plusieurs caches "proches" selon deux types de recherche : recherche par « tabou » qui parcourt les clients du réseau de façon aléatoire et la recherche séquentiel qui parcourt les clients du réseau par ordre de connexion.

Pour effectuer un tel architecture, on aura besoin d'utiliser deux catégories d'agent mobile : mono agent, si un client à la recherche d'un document, il va créer son propre agent et envoyer cet agent pour chercher le document dans un autre client et si l'agent ne trouve pas le document sur ce client, il va se déplacer vers un autre client et en suite multi-agent, dans cette catégorie, le client qui a reçu la demande du document si celui-là ne se trouve pas sur son cache, il va envoyer son propre agent vers un autre pour chercher le document du demandeur. L'architecture alors obtenue offre une grande flexibilité, permettant une configuration adaptée à un environnement donné, notamment en terme de coopération entre caches de requêtes.

Mots clés : Recherche de l'Information, Bases de Données MultiMedia Distribuées, Système d'Agents, caches coopératifs, Recherche Tabou, Recherche Séquentielle.

Sommaire

	Page
Liste des figures	5
Liste des tableaux	6
Liste des Abréviations et Acronymes	7
Introduction générale	8
Chapitre 1 : Base de données multimédias distribuées	10
Introduction	10
I. Base de données	11
1. Définition	11
2. Evolution des bases de données et de leur utilisation	11
2.1. Contexte	11
2.2. Modèles de données	12
2.2.1. Modèle relationnel.....	12
2.2.2. Modèle objet	13
2.2.3. Modèle relationnel-objet	14
II. Base de données distribuées	14
1. SGBD distribué	15
2. Problèmes à surmonter définis dans	16
3. Les réseaux sans fils	16
4. Traitement & Optimisation de Requêtes distribuées	17
5. Mise à jour des bases de données distribuée	17
III. Base de données multimédia	18
1. Multimédia	18
1.1. Ordres de grandeur	19
1.2. Chaîne du multimédia.....	20
1.3. Domaines d'application	20
2. Types de bases multimédia	23
3. Système de Gestion de Base de Données multimédia (SGBD-MM).....	23
3.1. Caractéristiques d'un SGBD multimédia	23
3.1.1. Base de données pour le multimédia.....	23
4. Architecture fonctionnelle d'un SGBD multimédia	24
Conclusion	24
Chapitre 2 : La gestion des Caches dans les réseaux mobiles	25

Introduction	25
I. Concepts et techniques de cache	25
1. Notion de cache	25
2. Déploiement d'un cache	26
2.1. Support physique d'un cache.....	26
2.2. Permissions d'accès	27
2.2.1. Généralités	27
2.2.2. Mécanisme d'autorisation	28
2.3. Place du cache	28
2.4. Entrée du cache	30
3. Fonctionnalités élémentaires d'un cache	31
3.1. Recherche au sein du cache	31
3.2. Remplacement d'entrées du cache	31
3.3. Résolution d'un défaut de cache	34
4. Fonctionnalités optionnelles d'un cache	34
4.1. Préchargement	34
4.2. Politique d'insertion et d'admission	35
4.3. Cache négatif	36
4.4. Politique d'allocation.....	36
5. Fonctionnalités gérées par des services externes au cache	36
5.1. Contrôle de concurrence	37
5.2. Synchronisation	37
II. Caches coopératifs	38
1. Les systèmes de caches coopératifs :	39
1.1. Geographical Push caching :	40
1.2. Adaptive caching :	41
1.3. Protocoles de Communication inter-caches :	42
1.3.1. Internet Cache Protocol (ICP) :	42
1.3.2. Cache digests : (Résumés des caches)	43
1.3.3. Summary cache :	43
1.3.4. Mécanismes de validation/réplication de documents :	44
1.3.5. Performances des systèmes de caches coopératifs :	45
III. La notion d'agent	46
1. Définition:	46
2. Les différentes multitudes de définitions d'agents	46
3. Les différentes catégories agents	47

4. Différence entre objet et agent	47
IV. Système multi-agent (SMA)	48
1. SMA est un système composé des éléments suivants	48
2. Architecture du SMA	48
Conclusion	49
Chapitre 3 : Conception et implémentation	50
Introduction	50
I. Les outils de développement	51
1. Le langage Java	51
2. La Plateforme JADE	51
2.1. Architecture logicielle de la plateforme JADE.....	52
2.2. L'interface de la plateforme JADE	53
2.3. Création d'un agent	54
2.3. Destruction d'un agent	54
II. La base de données de notre système de recherche	55
1. La base de données utilisée	55
2. Fichier trafic	55
3. Fichier cache	55
III. Modélisation	56
IV. Mécanismes de recherche	57
1. Recherche Tabou	57
2. Recherche séquentielle	57
V. Simulation	57
1. Interfaces principaux	57
1.1. Coté client	57
1.1.1. Menu fichier	59
1.1.2. Menu affichage	59
1.1.3. Menu journal et graphe	59
1.2. Coté serveur	59
1.2.1. Menu fichier	60
1.2.2. Menu affichage	61
2. Résultats de notre simulation.....	61
2.1. Taux de succès en requête (TSR)	61
2.2. Taux de succès en octet (TSO)	62
2.3. Le nombre de saut	62

2.4. Comparaison de deux types de recherches	63
Conclusion	63
Conclusion générale	64
Bibliographie	65

Liste des figures

<i>Figure 1.1</i> : réseau client/serveur trois tiers	17
<i>Figure1.2</i> : Audiovisuel.....	20
<i>Figure1.3</i> : biométrie.....	20
<i>Figure1.4</i> : commerce électronique.....	21
<i>Figure1.5</i> : rechercher texture spécifique pour l'industrie textile.....	21
<i>Figure1.6</i> : images satellitaires.....	22
<i>Figure1.7</i> : images botaniques.....	22
<i>Figure1.8</i> : images médicales.....	22
<i>Figure1.9</i> : Architecture fonctionnelle d'un SGBD multimédia.....	24
<i>Figure 2.1</i> : Fonctionnement d'un cache.....	26
<i>Figure 2.2</i> : Schéma de proxies caches coopératifs.....	39
<i>Figure 2.3</i> : Structure hiérarchique de proxy caches coopératifs.....	40
<i>Figure 2.4</i> : Structure en maille de proxy caches coopératives.....	41
<i>Figure 2.5</i> : Le protocole de communication inter cache ICP.....	42
<i>Figure. 3.2</i> : Architecture logicielle de la plate-forme JADE.....	52
<i>Figure. 3.3</i> : L'interface de la plateforme JADE.....	54
<i>Figure3.4</i> : Base de données utilisée.....	55
<i>Figure3.5</i> : Interface de client.....	57
<i>Figure3.6</i> : Menu fichier (coté client).....	59
<i>Figure3.7</i> : Menu affichage (coté client).....	59
<i>Figure3.8</i> : Menu journal et graphe	59
<i>Figure3.9</i> : Interface du serveur.....	60
<i>Figure3.10</i> : Menu fichier (coté serveur).....	60
<i>Figure3.11</i> : Menu affichage (coté serveur).....	61
<i>Figure3.12</i> : Taux de succès en requête.....	61
<i>Figure3.13</i> : Taux de succès en octet.....	59
<i>Figure3.14</i> : Le nombre de saut.....	63

Liste des tableaux

<i>Tableau 1.1</i> : Volume unitaire de stockage.....	12
---	----

Liste des Abréviations et des Acronymes

Abréviations	Signification
SGBD	Systèmes de Gestion de Base de Données
BD	Base de Données
ACL	Access Control List
ACE	Access Control Entries
SID	Security IDentifier
SGBD-MM	Système de Gestion de Base de Données Multimédia
FIFO	First In First Out
LIFO	Last In First Out
LRU	Least Recently Used
MRU	Most Recently Used
NFU	Not Frequently Used
NRU	Not Recently Used
LFU	Least Frequentlty Used
LFUPP	Least Frequentlty Used with Periodicity and Priority
ICP	Internet Cache Protocol
DNS	Domain Name System
CARP	Cache Array Routing Protocol
LARD	Locality Aware Request Distribution
LNC-R	Least Normalized Cost Replacement
GDS	Greedy Dual Size
LVCT	Least Value Based on Caching Time
LRFU	Least Recently/Frequently Used.
LNC-A	Least Normalized Cost Admission
TTL	Time To Live

Introduction générale

Durant la dernière décennie, la démocratisation des communications électroniques a entraîné l'essor des réseaux d'interconnexion. Composé à ses début de quelques machines dotés de liaison, internet s'est étendu, devenant présent partout et à tout moment. Outre l'étendue du réseau, ses capacités n'ont cessées de croître.

Les usages faits de cette possibilité accrue de communication ont, logiquement, suivi une évolution similaire. Débutant par la transmission de simple texte, le contenu des échanges s'est peu à peu enrichi. Des images sont venues agrémenter le texte, puis, la bande passante le permettant, l'échange de sons et de programme est apparu. De nos jours, il est courant de regarder une vidéo diffusée sur le réseau, et des offres proposant la télévision par internet apparaissent pour le grand public.

Cependant, la diffusion d'œuvre multimédia diffère de celle de données habituelles. Si le texte ou les images sont des informations de petite taille qui peuvent être téléchargées dans leur intégralité avant d'être affichées, il n'en est pas de même pour les films qui, par leur volume important, requerraient souvent plusieurs heures avant d'être visualisé. Une solution possible à ce surcharge est donc de décentraliser les données sous une forme de base de données distribuée et donc d'alléger les tâches du serveur et les liaisons entre ce serveur et les autres machines (terminaux), ce qui constitue le travail que nous allons réaliser.

Cependant nos outils sont des agents. L'idée de base est de donner la capacité à des agents de communiquer entre eux sur différentes machines. Ils s'échangent des données et informations à traiter, ainsi favorisent l'évolution considérable des systèmes repartis car ils permettent de réduire la surcharge des mobiles avec multiplateforme.

Le mémoire que nous allons défendre est que le déploiement d'ensemble de caches distribués coopératifs permet d'assurer l'extensibilité du système, d'optimiser la qualité du service offert aux utilisateurs et de supprimer les coupures dues à la mobilité.

Dans notre travail, on va vous présenter trois chapitres, premièrement on parlera sur les bases de données multimédia en abordant d'abord sur les bases de données en général avec des données multimédias puis nous verrons le concept de base de données multimédia distribuées,

Dans le deuxième chapitre, on parlera sur la gestion des caches dans les réseaux mobiles qui vont inclure un ensemble de politiques de gestion : une politique de suppression afin de supprimer les documents inintéressants, une politique d'insertion afin de décider des nouveaux documents à insérer et enfin une politique d'admission dont le rôle est de décider s'il dispose de ressources suffisantes pour accepter une nouvelle connexion. Un ensemble de politiques connus et pertinentes est présenté. Pour assurer l'extensibilité, un ensemble de caches doit être déployé. Afin d'optimiser cette architecture, des méthodes de coopération inter-caches ont été définies.

Dans le troisième chapitre, nous allons présenter l'environnement de développement, le modèle général du système à étudier, l'environnement d'implémentation ainsi que les résultats de l'implémentation, les mécanismes de recherche implémenté, et les critères de performances et enfin une comparaison entre les différents mécanismes de recherche.

Chapitre 1 : Base de données multimédias distribuées

Introduction

Au cours des dernières années, les bases de données ont connu un développement considérable, au point qu'elles jouent désormais un rôle dans chacune de nos opérations quotidiennes du simple achat effectué avec sa carte bancaire jusqu'à nos déclarations de revenus. Les révolutions technologiques intervenues dans le domaine des réseaux de communications ont permis à l'approche base de données distribuée de devenir une solution alternative à la centralisation. Les bases de données distribuées représentent un ensemble de bases stockées sur plusieurs ordinateurs, qui se comporte vis-à-vis des applications utilisatrices comme une base de données unique. Elles permettent de rassembler des données plus au moins hétérogènes au sein d'un réseau sous forme de base de données globale.

Les bases de données multimédias se trouvent à la croisée de l'évolution des bases de données et de l'émergence du multimédia en informatique où de nombreuses applications ont besoin de traiter des données non conventionnelles.

Les systèmes multimédias popularisés par des logiciels disponibles sur ordinateur personnel, permettent la manipulation de plusieurs types de données : textes, images, sons, vidéos. Ils offrent les fonctions de capture des données (caméra ou carte d'acquisition, par exemple), de construction et d'exécution de petits scénarios ou présentations multimédias combinant différents objets. Les objets multimédias sont en général stockés individuellement dans des fichiers traditionnels et l'un des challenges intéressant, consiste à combiner les fonctions de ces systèmes multimédias avec ceux des SGBDR. Il s'agit aussi de se placer non pas au niveau d'une informatique personnelle mais plutôt dans le contexte des systèmes distribués de grande taille combinant réseaux hauts débits et larges communautés d'utilisateurs répartis sur des sites distants. On arrive alors au concept de base de données multimédia distribuée dont les architectures physiques peuvent intégrer différentes sources (hétérogènes) de données mais qui doivent rester transparentes par rapport au poste client.

I. Base de données

1. Définition

Le nombre d'informations disponibles et les moyens de les diffuser sont en constante progression. La croissance du *World Wide Web* a encore accru ce développement, en fournissant l'accès à des bases de données très diverses avec une interface commune. Celles-ci se situent au cœur de l'activité des entreprises, des administrations, de la recherche et de bon nombre d'activités humaines désormais liées à l'informatique.

Dans le domaine purement informatique, elles interviennent dorénavant à tous les niveaux. Les développeurs d'applications s'appuient sur des bases de données externes pour gérer leurs données alors qu'auparavant elles étaient intégrées dans le programme. Citons un autre exemple : la gestion des fichiers dans les nouveaux systèmes d'exploitation (par exemple, Vista de Microsoft) évolue vers de véritables bases de données mises à jour en permanence. Elles permettent de retrouver les fichiers instantanément, par leur nom mais aussi par leur contenu, à la manière d'un moteur de recherche.

En résumé, on définit une base de données comme l'ensemble des données stockées.

Pour les manipuler, on utilise généralement un logiciel spécialisé appelé SGBD (*Système de Gestion de Bases de Données*). Il y a parfois confusion, par abus de langage, entre base de données et SGBD. On appelle aussi « système d'information » l'ensemble composé par les bases de données, le SGBD utilisé et les programmes associés. Plus formellement, on appelle **Base de Données** (BD) un ensemble de fichiers informatiques ou non structurés et organisés afin de stocker et de gérer de l'information [1].

2. Evolution des bases de données et de leur utilisation

2.1. Contexte

À partir des années 1960, les ordinateurs évoluent rapidement. Ils sont de plus en plus performants mais aussi de plus en plus répandus du fait de leur coût plus raisonnable. Leur utilisation change également ; on passe de la notion de calculateurs purs à des machines capables aussi de traiter de l'information. On parvient à un niveau d'abstraction supplémentaire par rapport aux machines et on obtient en conséquence une indépendance par rapport à l'architecture et surtout par rapport aux constructeurs.

La décennie des années 1970 est une période « faste » pour la recherche et l'innovation en informatique dont les résultats sont encore utilisés aujourd'hui. On peut utiliser des langages de programmation de haut niveau, afin de guider, voire de façonner, la démarche du programmeur (Pascal), et l'on envisage des systèmes d'exploitation indépendants de la machine employée (Unix). C'est également à cette époque que l'on pose les fondements des techniques qui sont utilisées dans les réseaux (TCP/IP), en particulier pour Internet. C'est dans ce contexte favorable que E. F. Codd définit et développe l'approche relationnelle en base de données.

L'objectif principal est d'éloigner l'utilisateur des détails d'implémentation et de faciliter ainsi l'usage de l'informatique. Un autre but est de rendre « **génériques** » et réutilisables les développements informatiques, parfois devenus caducs en raison d'un changement de machine. Dans le domaine des bases de données, le développement de l'**architecture à trois niveaux** constitue une première étape importante. Les fonctionnalités des systèmes de bases de données sont séparées en trois niveaux : *niveau physique, niveau logique et niveau externe*.

2.2. Modèles de données

Les modèles de données correspondent à la manière de structurer l'information dans une base de données. Ils reposent sur les principes et les théories issus du domaine de la recherche en informatique et permettent de traduire la réalité de l'information vers une représentation utilisable en informatique.

2.2.1. Modèle relationnel

En 1970, E. F. Codd propose un nouveau modèle « relationnel » dans [1] resté célèbre. Il cherche à créer un langage d'interrogation des bases de données plus proche du langage naturel. Dans ce modèle, les données sont stockées dans des tables, sans préjuger de la façon dont les informations sont stockées dans la machine. Un ensemble de données sera donc modélisé par un ensemble de tables. Le succès du modèle relationnel auprès des chercheurs, concepteurs et utilisateurs est dû à la puissance et à la simplicité de ces concepts. En outre, contrairement à certains autres modèles, il repose sur des bases théoriques solides, notamment la théorie des ensembles et la logique mathématique (théorie des prédicats d'ordre 1).

Les objectifs du modèle relationnel proposent des schémas de données faciles à utiliser, améliorent l'indépendance logique et physique, mettent à la disposition des

utilisateurs des langages de haut niveau pouvant éventuellement être utilisés par des non informaticiens, optimisent les accès à la base de données, améliorent l'intégrité et la confidentialité, fournissent une approche méthodologique dans la construction des schémas.

De façon informelle, on peut définir le modèle relationnel de la manière dont

Les données sont organisées sous forme de tables à deux dimensions, encore appelés relations et chaque ligne n-uplet ou tuple, les données sont manipulées par des opérateurs de l'algèbre relationnelle, l'état cohérent de la base est défini par un ensemble de contraintes d'intégrité.

Au modèle relationnel est associée la théorie de la normalisation des relations qui permet de se débarrasser des incohérences au moment de la conception d'une base de données [2].

2.2.2. Modèle objet

Dans le sillage du développement des langages orientés objet (C++, Java...) dans les années 1980, le **concept objet** a été adapté aux bases de données. Plusieurs raisons, en dehors des qualités reconnues de l'approche objet, ont conduit à définir une extension objet pour les bases de données. La première est que le modèle relationnel, dans sa simplicité, ne permet pas de modéliser facilement toutes les réalités. La deuxième est qu'un objet permet de représenter directement un élément du monde réel. Les structures d'éléments complexes se retrouvent souvent dispersées entre plusieurs tables dans l'approche relationnelle classique. De plus, le

concept objet est mieux adapté pour modéliser des volumes de texte importants ou d'autres types de données multimédias (sons, images, vidéos...). Enfin, il est beaucoup plus commode de manipuler directement des objets lorsque l'on développe avec un langage à objet (comme C++ ou Java). Les bases de données, on le rappelle, sont dorénavant des briques constitutives des applications. Les **bases de données « orientées objet »** apportent ainsi aux applications développées en langage objet la **persistance** des objets manipulés : ces derniers peuvent ainsi directement être réutilisés par l'application d'origine ou par d'autres sans redéfinition. Ces concepts ont été intégrés à partir de la version 2 de la norme SQL.

Les produits commerciaux adaptés à ces concepts n'ont pas connu une diffusion suffisamment importante. Le monde des bases de données évolue assez lentement : la migration d'un système d'information vers l'objet représente pour une organisation un investissement considérable qui n'est pas toujours justifié. La robustesse et la popularité de l'approche relationnelle, qui a mis presque vingt ans à s'imposer, a également freiné le développement de l'approche objet pure dans les bases de données. Les données modélisées sous forme d'objets

sont aussi plus complexes à représenter du point de vue du SGBD et l'on rencontre encore très souvent des problèmes de performance.

2.2.3. Modèle relationnel-objet

Une demande d'évolution du strict modèle relationnel existe toutefois. En effet, la gestion des données autres que du texte et des nombres comme des images, du son et des vidéos implique l'évolution du modèle relationnel. De même, les champs dits « multivalués », disposant de plusieurs valeurs telles qu'une liste de prénoms ou des coordonnées géographiques, ne peuvent pas être modélisés efficacement en utilisant ce type d'approche. L'idée est alors d'intégrer de l'objet au modèle relationnel existant plutôt que d'utiliser l'approche objet pure. Il convient de remarquer que ce type d'évolution a déjà été développé dans le cadre des langages de programmation. Le langage C++ est l'évolution intégrant l'approche objet du langage C et non pas un langage à objet pur comme peut l'être Small talk.

Cette extension, adoptée par la plupart des SGBD, se nomme « **relationnel-objet** » et permet aux concepteurs des bases de données de disposer de types évolués « abstraits » plus simples à concevoir et surtout plus commodes à faire évoluer. Elle offre en outre la possibilité de modéliser plus facilement la complexité des organisations. Dans cette optique, la norme SQL a logiquement été adaptée. Dans sa version 3, elle prend en compte l'extension objet. Les types de données sont étendus et les opérations d'encapsulation et d'héritage, typiques de l'approche objet, sont supportées. Cette solution a l'avantage d'offrir un bon niveau de compatibilité avec l'approche précédente très répandue et d'effectuer ainsi une migration plus aisée.

II. Base de données distribuées

Le déploiement des réseaux ainsi que l'augmentation de leur débit ces dernières années ont conduit à répartir les données sur plusieurs sites géographiques, ce qui facilite la politique de décentralisation des organisations. Ce type d'architecture masque la répartition de l'information tout en garantissant une gestion transparente aux utilisateurs, comme s'ils disposaient d'une seule base de données. Les bases de données distribuées assurent ainsi une *plus grande fiabilité*, de *meilleures performances* et facilitent *l'évolution du système d'information*.

- ✓ **La fiabilité et la sécurité.** On effectue une copie de sécurité des données sur un site distant à intervalles réguliers pour éviter le désastre de la perte de données due à un incendie par exemple.
- ✓ **La disponibilité.** On procède à des répliques quasi permanentes des données dans le but de « rapprocher » les utilisateurs des données d'un point de vue de la topologie du réseau. On améliore également le temps de réponse en répartissant la charge entre les serveurs. Cette distribution est gérée de manière intelligente par les systèmes informatiques, ce qui permet de répartir l'information efficacement sur les différents sites, en fonction des accès utilisateurs. Ces principes sont notamment utilisés par les moteurs de recherche.
- ✓ **Les données sont distribuées sur des sites séparés.** Le dispositif permet de masquer cet aspect aux utilisateurs et de fonctionner comme si un seul serveur était présent sur un seul site. L'évolution du système est rendue totalement **transparente** pour les utilisateurs : en cas notamment de changement de machine à la suite d'une panne, de modification de localisation, d'augmentation de la taille de la base, d'ajouts d'ordinateurs sur le réseau afin d'augmenter la capacité de stockage de la base de données.

Ces technologies possèdent néanmoins des inconvénients. La sécurité sur les réseaux informatiques nécessite beaucoup plus de travail que dans le cas d'un système non réparti.

Les techniques de sécurité à mettre en œuvre sont plus complexes et plus coûteuses.

1. SGBD distribué

Une base de données centralisée est gérée par un seul SGBD, est stockée dans sa totalité à un emplacement physique unique et ses divers traitements sont confiés à une seule et même unité de traitement. Par opposition, une base de données distribuée [3] est gérée par plusieurs processeurs, sites ou SGBD. Un système de bases de données distribuées ne doit donc en aucun cas être confondu avec un système dans lequel les bases de données sont accessibles à distance. Il ne doit non plus être confondu avec une multibase ou une BD fédérée.

Dans une multibase, plusieurs BDs interopèrent avec une application via un langage commun et sans modèle commun.

Dans une BD fédérée, plusieurs BDs hétérogènes sont accédées comme une seule via une vue commune.

Du point de vue organisationnel nous distinguons deux architectures :

- ✓ *Architecture Client-Serveur* : les serveurs, ont pour rôle de servir les clients.

Par servir, on désigne la réalisation d'une tâche demandée par le client.

- ✓ *Architecture Pair-à-Pair (Peer-to-Peer, P2P)* : par ce terme on désigne un type de communication pour lequel toutes les machines ont une importance équivalente.

2. Problèmes à surmonter définis dans [4]

- ✓ **Coût** : la distribution entraîne des coûts supplémentaires en termes de communication, et en gestion des communications (hardware et software à installer pour gérer les communications et la distribution).
- ✓ **Problème de concurrence**.
- ✓ **Sécurité** : la sécurité est un problème plus complexe dans le cas des bases de données réparties que dans le cas des bases de données centralisées.

3. Les réseaux sans fils

Un réseau sans fils (en anglais Wireless network) est, comme son nom l'indique, un réseau dans lequel au moins deux terminaux peuvent communiquer sans liaison filaire. Grâce aux réseaux sans fils, un utilisateur a la possibilité de rester connecté tout en se déplaçant dans un périmètre géographique plus ou moins étendu, c'est la raison pour laquelle on entend parfois parler de "mobilité".

Les réseaux sans fils sont basés sur une liaison utilisant des ondes radio-électriques (radio et infrarouges) en lieu et place des câbles habituels. Il existe plusieurs technologies se distinguant d'une part par la fréquence d'émission utilisée ainsi que le débit et la portée des transmissions [5].

Les réseaux sans fils permettent de relier très facilement des équipements distants d'une dizaine de mètres à quelques kilomètres. De plus l'installation de tels réseaux ne demande pas de lourds aménagements des infrastructures existantes comme c'est le cas avec les réseaux filaires. En contrepartie se pose le problème de la réglementation relative aux transmissions radio-électriques. De plus les ondes hertziennes sont difficiles à confiner dans une surface géographique restreinte, il est donc facile pour un pirate d'écouter le réseau si les informations

circulent en clair. Il est donc nécessaire de mettre en place les dispositions nécessaires de telle manière à assurer une confidentialité des données circulant sur les réseaux sans fils.

Soit le réseau client/serveur trois tiers sur la **Figure 1.1** :

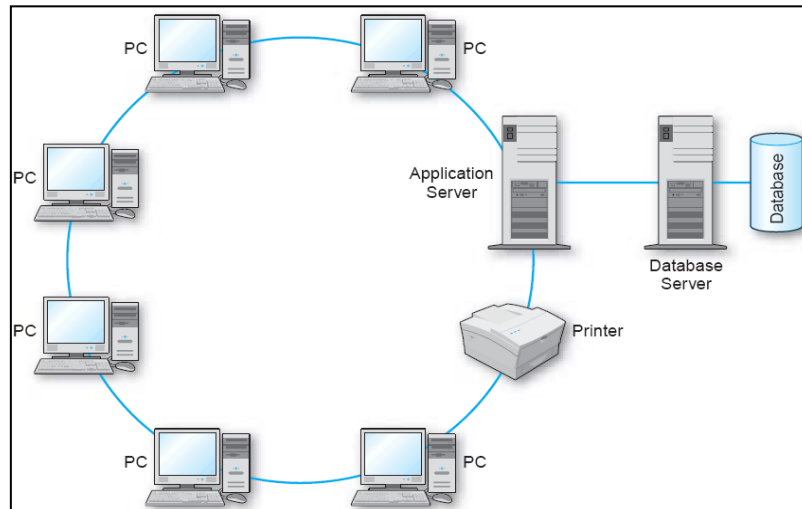


Figure 1.1: réseau client/serveur trois tiers [6]

Considérons que ce réseau est destiné à être utilisé par une firme multinationale, dont les principaux sites sont à Los-Angeles, Memphis, New-York (lequel sont les quartiers généraux de l'entreprise), Paris et Tokyo. Supposons qu'il existe beaucoup de transactions au niveau des cinq sites, notre base de données est composée de six tables qui sont A, B, C, D, E, et F. Nous signalons que le et temps de la réponse pour une requête sur la base de données est un facteur très important.

5. Traitement & Optimisation de Requêtes distribuées

Les règles d'exécution et les méthodes d'optimisation de requêtes définies pour un contexte centralisé sont toujours valables, mais il faut prendre en compte d'une part la fragmentation et la répartition des données sur différents sites et d'autre part le problème du coût des communications entre sites pour transférer les données. Le problème de la fragmentation avec ou sans duplication concerne principalement les mises à jours tandis que le problème des coûts des communications concerne surtout les requêtes.

6. Mise à jour des bases de données distribuée

La principale difficulté réside dans le fait qu'une mise à jour dans une relation du schéma global se traduit par plusieurs mises à jour dans différents fragments. Il faut donc

identifier les fragments concernés par l'opération de mise à jour, puis décomposer en conséquence l'opération en un ensemble d'opération de mise à jour sur ces fragments.

Insertion

Retrouver le fragment horizontal concerné en utilisant les conditions qui définissent les fragments horizontaux, puis insertion du tuple dans tous les fragments verticaux correspondants.

Suppression

Rechercher le tuple dans les fragments qui sont susceptibles de contenir le tuple concerné, et supprimer les valeurs d'attribut du tuple dans tous les fragments verticaux.

Modification

Rechercher les tuples, les modifier et les déplacer vers les bons fragments si nécessaire.

III. Base de données multimédia

Une **base de données** est un ensemble structuré et organisé permettant le stockage de grandes quantités d'informations afin d'en faciliter l'exploitation (ajout, mise à jour, recherche de données).

1. Multimédia

Le media est tout support de diffusion de l'information constituant à la fois un moyen d'expression et un intermédiaire transmettant un message à l'attention d'un groupe.

Le mot multimédia est apparu vers la fin des années 1980, lorsque les CD-ROM se sont développés. Il désignait alors les applications qui, grâce à la mémoire du CD et aux capacités de l'ordinateur, pouvaient générer, utiliser ou piloter différents médias simultanément [7]

Aujourd'hui on utilise le mot multimédia pour désigner toute application utilisant ou servant à travailler sur au moins un média spécifique.

Par ailleurs, en recherche en informatique, on nomme multimédia l'étude des médias non textuels, principalement les images, les vidéo et les sons.

La définition de travail des contenus pouvant être de différents types, en particulier Texte, Image, Son, Vidéo peuvent être combinés et Caractérisés par des besoins de stockage très importants.

1.1. Ordres de grandeur

- ✓ Volume unitaire de stockage. Valeurs typiques :

	Stockage brut	Compression sans perte	Compression avec perte
Texte (Descarte, discours de la méthode, http://abu.cnam.fr)	125Ko	Gzip : 42 Ko	N/A
Photo (6 Mpixels)	18 Mo	Gzip* : 12 Mo	1 Mo
Album de musique	750 Mo	FLAC : 406 Mo	MP3 : 75 Mo
Un film de 100 minutes (720x576)	180 Mo	x	500 Mo

Tableau 1.1 : Volume unitaire de stockage

- ✓ Nombre d'unités
 - Internet visible : nécessite "seulement" 5-10 To de capacité de stockage
 - ❖ Sept 2011: > 1 milliards de recherches / jour
 - Films : <http://www.imdb.org> recense plus de 400 000 films
 - Images (semi-)pro : Corbis, Getty, Fotolia = 10-100 M images
- ✓ Croissance très importante, en raison de l'accumulation des contenus numériques autoproduits par le grand public
 - Images : par exemple, Facebook
 - ❖ ordre de grandeur: 10-100 milliards d'images
 - ❖ Facebook + 100 M d'images / jour
 - <http://www.flickr.com> : "9600 images uploaded in the last minute"
 - ❖ extrapolation : 14M d'images par jour, 5.2G d'images par an
 - <http://www.youtube.com> : 2G de vidéos délivrées chaque jour + 24 h de vidéo / minute
 - croissance > loi de Moore → "datacenters"
- ✓ Images/vidéo spécifiques : photos satellites, vidéo-surveillance (4 M en GB), etc.

1.2. Chaîne du multimédia

Les différents besoins sont :

- ✓ Génération : outils de production et de création
- ✓ Représentation : utilisation de formats de représentations différentes
- ✓ Stockage
- ✓ Transmission : problème de réseaux, architecture
- ✓ **Recherche d'information : recherche basée sur le contenu**
- ✓ Distribution : conception de serveur de streaming

1.3. Domaines d'application

- ✓ **Audiovisuel** : par exemples : détection de copies (droits), retrouver un plan spécifique d'un programme, annotation automatique de vidéos



Figure 1.2 : Audiovisuel

- ✓ **Sécurité** : par exemples : biométrie (empreintes, iris), vidéosurveillance

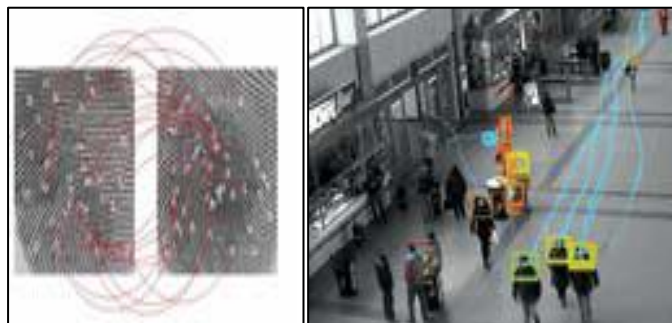


Figure 1.3 : biométrie

- ✓ **Internet** : par exemples : commerce électronique, annotation et filtrage Web2.0



Figure1.4 : commerce électronique

- ✓ **Design, publicité, architecture** : par exemples : rechercher texture spécifique pour l'industrie textile, illustrer une publicité par une photo adéquate, détecter des tendances



Figure1.5: rechercher texture spécifique pour l'industrie textile

- ✓ **Art, éducation** : par exemples : recherche encyclopédique d'illustrations, d'une œuvre d'art spécifique.
- ✓ **Bases d'images satellitaires** : par exemples : suivre l'évolution de la végétation, évaluer l'impact des feux de forêt

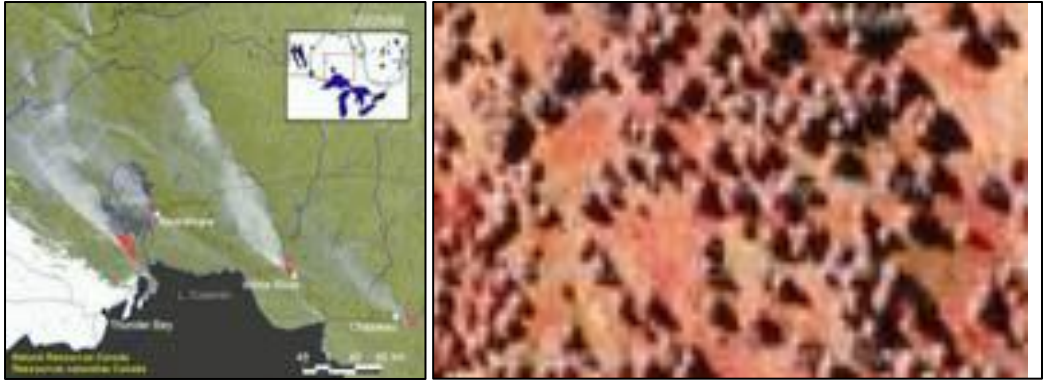


Figure1.6 : images satellitaires

- ✓ **Bases d'images botaniques :** par exemple : identifier les gènes qui interviennent dans une même chaîne de synthèse protéinique



Figure1.7 : images botaniques

- ✓ **Bases d'images médicales :** par exemple : quelles images indiquent une pathologie (but éducatif ou diagnostic)

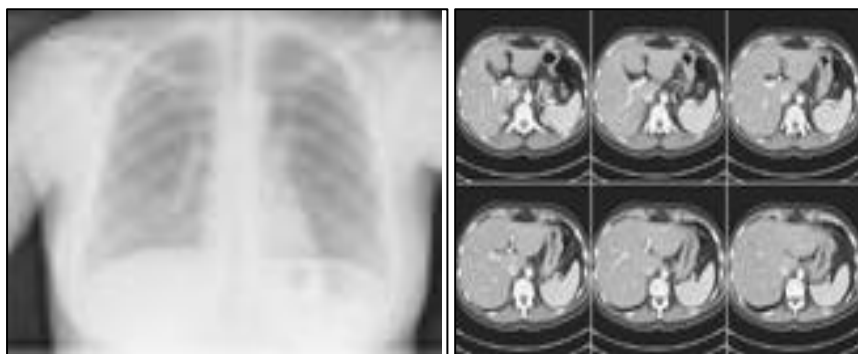


Figure1.8 : images médicales

2. Types de bases multimédia

- ✓ **Bases génériques** : sont des contenu hétérogène (bases grand public, Internet, archives généralistes) en général sans vérité-terrain naturelle
- ✓ **Bases spécifiques** : sont spécifiques à un domaine d'application dont le contenu homogène (visages, empreintes digitales, monnaies...) est en général avec vérité-terrain naturelle

3. Système de Gestion de Base de Données multimédia (SGBD-MM)

3.1. Caractéristiques d'un SGBD multimédia

Une base de données multimédia contient des informations conservées sous différents formats multimédias : fichiers sons, fichiers photo, séquence vidéo, données textes etc... Le volume constitue une des principales caractéristiques de ces données qui peuvent atteindre quelques giga-octets. A la lumière des caractéristiques que nous venons de citer, un SGBD multimédias doit pouvoir :

- ✓ Stocker des objets de gros volumes de données.
- ✓ Offrir des outils de modélisation (modèle, langage de définition de données) pour prendre en compte la nature spécifique des objets et les opérations associées.
- ✓ Mettre à la disposition des usagers des langages et des interfaces spécifiques pour la manipulation et l'interrogation de ces données.

Optimiser les accès aux données en offrant des outils d'indexation spécialisés.

3.1.1. Base de données pour le multimédia

- ✓ Initialement traitées comme des bases standards est objets multimédia traités comme un seul item comme champs au sein d'une base de données relationnelle (ex: Oracle) et objet opaque
- ✓ Recherche sur mots clés introduits manuellement dans le système par la personne : nécessité d'un système d'annotation, par exemple : Tag ID3 dans le MP3 est utilisation des relations entre objets et la recherche sur les mots présents dans les pages Web conjointement avec l'objet multimédia comme par exemple : <http://images.google.com>, <http://www.exalead.fr/image>

4. Architecture fonctionnelle d'un SGBD multimédia

La figure 1.9 présente une architecture fonctionnelle en couches pour un SGBD Multimédia. Au plus bas niveau figure un serveur d'objets qui prend en compte les aspects

structurels et le volume des données. Généralement un SGBD classique peut servir à réaliser cette couche mais il doit être étendu si l'on veut gérer les différents aspects des données multimédias qui ont été décrits plus haut. Cette extension constitue une couche logicielle qui se place au-dessus du serveur d'objets et qui est spécialement conçue pour traiter les aspects spatiaux et temporels, par exemple. Ainsi elle permet la composition et la présentation d'objets qui sont gérés de manière naturelle au niveau de l'interface.

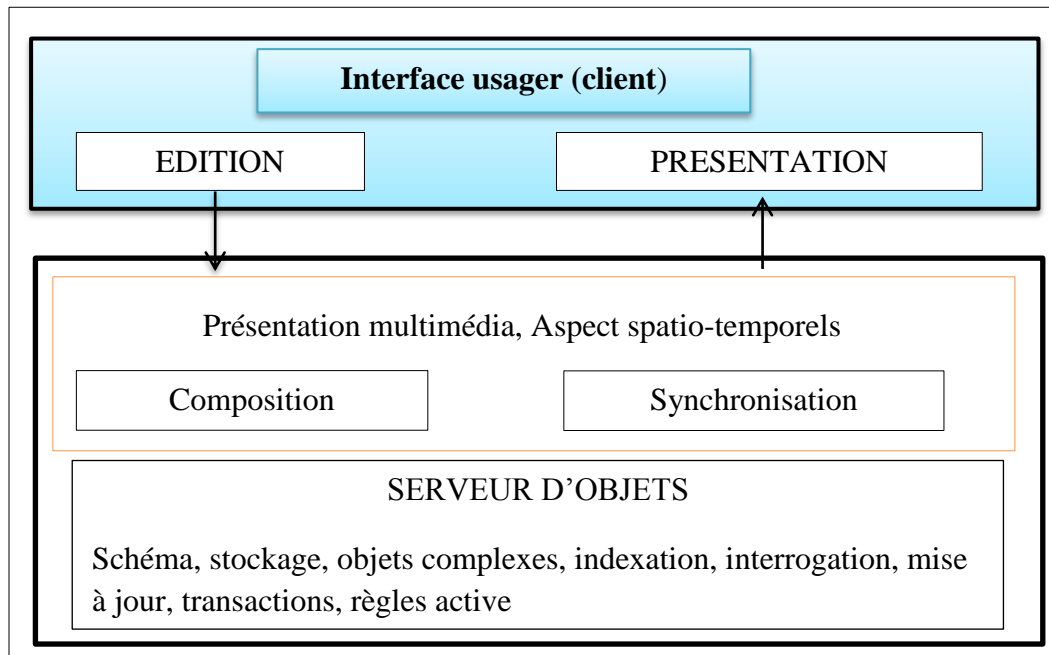


Figure 1.9 : Architecture fonctionnelle d'un SGBD multimédia

Conclusion

L'évolution récente de l'utilisation des données multimédias dans les applications a été phénoménale. Les bases de données multimédia sont essentielles pour une gestion efficace et l'utilisation efficace des énormes quantités de données multimédias. La diversité des applications utilisant des données multimédia, la technologie évolue rapidement, et les complexités inhérentes à la représentation, l'interprétation sémantique et la comparaison de la similarité posent de nombreux défis. Les SGBD-MM sont encore à leurs balbutiements, et sont étroitement liées à réduire les domaines d'application. Les expériences acquises à l'élaboration et l'utilisation de nouvelles applications multimédias contribueront à faire progresser la technologie de base de données multimédia.

Chapitre 2 : La gestion des Caches dans les réseaux mobiles

Introduction

L'objectif de ce chapitre est de définir la notion de cache et les concepts associés, avant de présenter les caractéristiques considérées au sein d'un cache afin de mieux comprendre la difficulté de proposer un service de caches.

Ce chapitre est composé de deux grandes sections. La première est le Concepts et techniques de cache qui est organisée de la manière suivante. La sous-section 1 présente le concept de cache. La sous-section 2 s'intéresse aux caractéristiques d'un cache liées au déploiement. Les fonctionnalités élémentaires et optionnelles d'un cache sont alors étudiées dans les sous sections 3 et 4. La sous-section 5 s'intéresse quant à elle aux fonctionnalités utilisées par un cache mais gérées par d'autres services. La deuxième section sera les caches coopératifs qui a pour objectif de présenter plus en détails les caches coopératifs, visant à optimiser l'utilisation des ressources globales. Finalement, il se terminera par une conclusion.

I. Concepts et techniques de cache

1. Notion de cache

Les caches sont des espaces physiques limités permettant de stocker des copies d'objets autorisant des accès plus rapides pour de futures utilisations. Le principe d'un cache consiste à conserver des copies des objets demandés par des utilisateurs ou des applications, évitant ainsi de contacter les serveurs si ces objets sont de nouveaux référencés.

L'utilisation d'un **cache** a toujours été importante dans les systèmes informatiques, afin de fournir une haute qualité de service et notamment une amélioration des temps de réponse perçus par les utilisateurs. En effet, un cache fournit un accès rapide aux données, pouvant tirer bénéfice de l'utilisation d'un support physique efficace, d'un ensemble d'éléments gérés plus petit autorisant des recherches plus rapides, ou encore d'un placement évitant des communications réseaux. De plus, employer un cache réduit la charge sur les serveurs et éventuellement sur les liens de communication, la récupération des éléments étant alors plus rapide. Ainsi un cache améliore généralement les performances, en dépit des coûts de déploiement, d'administration, de gestion et éventuellement des problèmes de cohérence qu'il peut engendrer.

Le fonctionnement classique d'un cache est illustré par la figure 2.1. A la suite d'une demande d'un client (1), un cache vérifie la présence de l'élément dans son espace de stockage. Si celui-ci est contenu en cache (**succès de cache** ou *cache hit*), il est fourni directement au client (2). Dans le cas contraire (**défaut de cache** ou *cache miss*), il est nécessaire de contacter le serveur (2') afin de le récupérer (3'). Le cache renvoie l'élément (4') au client et le stocke pour une utilisation future. A noter qu'en général, l'utilisation du cache est transparente, le client n'ayant pas conscience qu'il accède aux données au travers d'un cache. Cependant, la transparence est parfois levée, comme par exemple dans [8], permettant notamment à un client de contacter directement le serveur s'il veut s'assurer d'obtenir la dernière version d'un élément.

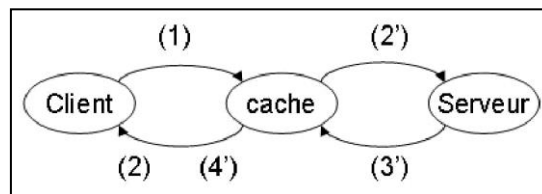


Figure 2.1 : Fonctionnement d'un cache

- ✓ Un succès de cache (*cache hit*) se produit, lorsque l'élément demandé est présent en cache.
- ✓ Un défaut de cache (*cache miss*) se produit, lorsque l'élément demandé n'est pas présent en cache.

2. Déploiement d'un cache

Cette section présente les caractéristiques liées au déploiement d'un cache : le support physique, la taille, le droit de modification et la place d'un cache.

2.1. Support physique d'un cache

Le support physique correspond au matériel utilisé pour stocker les différents éléments placés en cache.

Le concept de cache est souvent associé à la notion de mémoire. En effet, il s'agit du premier support utilisé et de nos jours, un très grand nombre de caches reposent sur ce support physique, comme dans le système de fichiers **Sprite** [9] par exemple. Néanmoins, il convient de noter que des caches **disques** sont proposés dans différentes applications, comme les systèmes de fichiers, les systèmes de gestion de base de données client-serveur ou encore

dans la plupart des caches Internet. En stockant les derniers blocs utilisés, le cache disque permet :

- ✓ de diminuer le nombre d'entrée/sortie,
- ✓ de réaliser des écritures asynchrones.

La présence d'un cache disque pose le problème de la cohérence des informations sur disque.

Ce problème est réglé par la mise à jour périodique du disque (par exemple toutes les 30 secondes pour UNIX ou immédiatement pour MS-DOS). Alors qu'un cache mémoire offre un accès aux données plus rapide, un disque quant à lui fournit un espace de stockage plus important et surtout une persistance du contenu. Bien sûr, ces deux solutions peuvent être combinées, où un cache mémoire est utilisé en premier lieu, avec traitement de la demande par un cache disque si nécessaire.

2.2. Permissions d'accès

2.2.1. Généralités

Un mécanisme de contrôle d'accès assure les fonctions d'authentification, d'autorisation et de journalisation lors de l'accès d'une entité à une ressource. Le contrôle d'accès discrétionnaire est essentiellement basé sur la notion de propriété. Chaque ressource est protégée individuellement, selon des autorisations placées par le propriétaire de l'objet vis-à-vis d'autres entités du système. Le propriétaire initial est le plus souvent le créateur de l'objet, mais la propriété peut être transférée. Ce transfert peut être manuel, ou bien automatique comme dans le cas d'ajouts d'éléments dans l'annuaire par un administrateur, dont le propriétaire devient le groupe Admins de domaine et non plus le créateur. Toute ressource protégeable par le contrôle d'accès discrétionnaire Microsoft est appelée « Securable Object ». Quel que soient son type et son mode de stockage, elle dispose systématiquement d'un descripteur de sécurité qui lui est propre. Les objets du système de fichiers, les clés de la base de registre, les objets de l'Active Directory sont autant d'exemples de « Securable Objects ». Un descripteur de sécurité contient notamment la liste de contrôle d'accès (ACL) composée d'entrées décrivant chaque permission accordée ou refusée : les ACE. Toute entité susceptible d'être titulaire d'une permission (utilisateur, machine, etc.) est appelée « Security Principal ». Dans le modèle Microsoft, elle est désignée par un SID 1.

2.2.2. Mécanisme d'autorisation

Le plus souvent, les ressources sont créées dans un conteneur, au sein d'une hiérarchie correspondant à leur contexte et à leur type, par exemple une arborescence de répertoires dans un système de fichiers ou la structure de la base de registre. La hiérarchie se traduit en un concept d'héritage du contrôle d'accès. Les permissions appliquées aux éléments parents vont généralement être incluses dans le descripteur de sécurité des éléments enfants lors de leur création, ou lors de la modification du descripteur du parent. L'héritage peut être bloqué sur l'élément enfant par un paramétrage de ce dernier ou si l'ACE est marquée comme non héritable. Le propriétaire d'un objet dans le modèle Microsoft est appelé « primaryOwner » et possède par défaut les privilèges lui permettant de lire et de modifier le descripteur de sécurité. Par conséquent, le propriétaire peut tout à fait ne pas disposer de permissions apparentes sur une ressource et se les accorder au besoin. La vérification de l'autorisation [8] s'effectue de manière séquentielle en suivant la liste de contrôle d'accès portée par une ressource via son descripteur de sécurité, selon les droits demandés. Un processus demandeur présente généralement plusieurs SID dans son jeton d'accès, comme celui de l'utilisateur sous l'identité duquel le processus tourne et ceux des groupes auxquels cet utilisateur appartient : seules les ACE dont le SID titulaire est présent et actif dans le jeton d'accès du demandeur sont conservées; chaque ACE restante est considérée à son tour et valide individuellement les bits de l'Access Mask demandé, à partir de son propre Access Mask ; le mécanisme s'arrête dès que tous les bits de l'Access Mask demandé sont validés, ce qui autorise l'accès. Si le mécanisme prend fin parce que toutes les ACE ont été inspectées, l'accès est refusé; – si une ACE d'interdiction valide un bit demandé, l'accès est refusé immédiatement.

Les entrées d'un cache peuvent être accédées en lecture ou en écriture. Les **permissions d'accès** caractérisent les opérations autorisées sur les éléments stockés en cache. Elles peuvent être globales au cache, mais également liées aux éléments ou aux clients. Autoriser les modifications au sein d'un cache engendre une synchronisation plus complexe des différentes copies que pour une approche lecture seule. Néanmoins, cette approche fournit une plus grande autonomie, évitant par exemple des communications coûteuses dans les environnements mobiles.

2.3. Place du cache

Dans le cas des applications réparties, un cache peut être exécuté sur un client (cache client), sur un serveur (cache serveur) ou encore sur une machine intermédiaire (cache intermédiaire).

Avec un *cache client*, chaque utilisateur ou application dispose de son propre cache. Ainsi, la charge sur les serveurs et l'utilisation de la bande passante sont plus faibles puisque une partie des demandes est traitée localement. En cas de panne des serveurs, ou d'une déconnexion, un *cache client* fournit même une certaine disponibilité, les éléments ayant été chargés et stockés pouvant être utilisés. Cette approche est particulièrement intéressante pour le passage à l'échelle, les ressources étant alors proportionnelles au nombre de clients. De plus, le cache n'étant utilisé que par un unique client, il est possible d'avoir une idée précise des éléments devant être conservés en cache.

Contrairement à un cache client, un *cache serveur* peut être partagé entre différents utilisateurs ou différentes applications. Dans ce cas, une seule copie est stockée en cache, permettant d'optimiser l'utilisation de l'espace physique global (contrairement à un cache client, où dans le pire des cas, le nombre de copies peut être égal au nombre de clients) et facilitant la gestion des mises à jour. Le partage du cache permet aussi de mieux caractériser les éléments les plus référencés globalement et éventuellement d'anticiper leurs demandes.

Avec un *cache intermédiaire*, le cache est associé à un serveur applicatif particulier, déployé sur une machine dédiée. Des caches intermédiaires ont ainsi été proposés dans les systèmes de médiations tel que **SIMS** [23], ou encore dans des serveurs mandataires pour Internet comme **Relai** [Relay] [24] ou le serveur mandataire du Centre Européen de Recherches Nucléaires. A noter que le service applicatif est parfois uniquement consacré au cache. Bien que par leur caractère souvent organisationnel, les caches intermédiaires soient généralement déployés du côté des clients, ils arrivent qu'ils soient placés à proximité des serveurs. On parle dans ce cas de **serveur mandataire inversé** (*reverse proxy*). Un cache intermédiaire peut être vu comme un compromis intéressant entre un cache client et un cache serveur. Comme un cache client, il diminue la charge sur le serveur et, s'il est placé proche des clients, réduit les communications et l'utilisation de la bande passante. Comme un cache serveur, il partage son contenu pour différents clients, optimisant ainsi l'espace physique disponible et permettant de déterminer les éléments les plus globalement utilisés, autorisant l'intégration d'algorithmes efficaces de prédiction. Contrairement à un cache serveur pour lequel les demandes peuvent diverger et ainsi dégrader les performances, un cache intermédiaire est souvent placé à un niveau institutionnel, les utilisateurs étant susceptibles d'accéder à un même sous ensemble d'éléments. Enfin, à l'instar d'un cache client, un cache intermédiaire offre une certaine disponibilité.

2.4. Entrée du cache

Les **entrées du cache** correspondent aux éléments manipulés au sein du cache. Le type d'une entrée dépend du contexte applicatif et en particulier de l'architecture utilisée. Ainsi dans les systèmes de gestion de bases de données, les éléments en cache sont souvent des pages ou des objets (correspondant à des n-uplets).

Dans les caches d'objets [11], les entrées manipulées sont des objets (des n-uplets dans le cas des systèmes de gestion de bases de données). Un cache d'objets est ainsi insensible au regroupement des données effectué du côté du serveur, puisque les accès se font en terme d'objets et non en terme de groupes comme pour les caches de pages. Ceci permet notamment une utilisation de l'espace physique plus efficace, puisque tous les objets présents en cache correspondent à des objets référencés par l'utilisateur ou l'application (sauf dans le cas d'un chargement anticipé). Le fait de ne mettre en cache que des objets référencés optimise également la consommation de la bande passante. Enfin, la gestion de la concurrence est plus fine, puisque le verrouillage peut se faire au niveau des objets, évitant de bloquer l'accès à des éléments non utilisés à un moment donné.

Avec un cache de pages, les transferts depuis le serveur se font en termes de pages disque. Avec cette approche, le coût d'accès des éléments en cache est réduit, une page représentant une agrégation d'objets. Un regroupement efficace des objets permet de plus d'anticiper de futures demandes des clients. En effet, si les pages rassemblent les objets qui ont une forte probabilité d'être accédés dans un même intervalle de temps, alors la récupération de la page associée à un objet demandé permettra d'éviter de contacter le serveur pour obtenir les autres objets qu'elle contient. A noter que dans les systèmes de fichiers pour grille, le *bloc*, tout comme une page, représente une unité de transfert de taille fixe. Contrairement à une page qui regroupe différents objets, un bloc permet le découpage des fichiers dont la taille trop importante pourrait dégrader les performances de l'application.

Dans les systèmes d'interrogation de données, les accès aux éléments du cache peuvent se faire en termes de *résultats de requêtes*, une entrée associant alors une requête aux objets réponses, rendant la recherche en cache ou la récupération de données sur le serveur moins coûteuses. Contrairement à une page, la taille des résultats de requêtes est variable. Ce mécanisme permet en particulier de ne pas garder des objets non référencés en cache et d'optimiser l'utilisation de la bande passante.

3. Fonctionnalités élémentaires d'un cache

Le fonctionnement d'un cache repose sur un ensemble de fonctionnalités dont certaines sont nécessaires pour fournir le service souhaité et d'autres optionnelles pour améliorer les performances obtenues. Dans cette section, nous nous intéressons aux fonctionnalités élémentaires d'un cache : la recherche, le remplacement et la résolution.

3.1. Recherche au sein du cache

Lorsqu'une demande est posée, l'élément correspondant est recherché au sein du cache. Puisque la recherche d'un élément est un événement fréquent, le cache doit fournir des méthodes efficaces pour fournir cette stratégie. Cette fonctionnalité est capturée dans la *politique de recherche* du cache.

La politique de recherche est fortement liée à la structure de données utilisée. Le choix d'une structure de données dépend d'un compromis entre la complexité de gestion des données et l'efficacité des recherches. Ainsi en fonction du contexte applicatif, des structures simples nécessitant des parcours séquentiels, tels que des listes, des tableaux ou des structures plus complexes offrant des accès plus performants, à l'instar des tables de hachage, peuvent être employées.

3.2. Remplacement d'entrées du cache

Contrairement au système de réplication qui fournissent l'ensemble des objets disponibles en plusieurs exemplaires, un cache possède par définition un espace de stockage limité, dont la taille est inférieure à celle formée par l'ensemble des documents disponibles, et ne peut donc emmagasiner tous les objets qui ont été et seront demandés. Dès lors, un mécanisme permettant de renouveler le contenu entreposé au cours du temps doit être mis en place, afin de rester en phase avec les demandes des clients. Faute de quoi, le cache deviendra inutile car entièrement rempli de données qui ne seront plus utilisées. Afin de permettre ce renouvellement, de l'espace doit être libéré évinçant les documents les moins propices au maintien. Cette tâche est dévolue aux politiques de suppression.

Il n'existe pas de politique de remplacement meilleure que les autres pour toutes les séquences d'accès. C'est pourquoi, de nombreuses solutions sont proposées dans la littérature. Le choix d'une politique est fortement lié à l'application et aux objectifs fixés par le développeur de la solution de cache. Les politiques de remplacement peuvent, en particulier, s'appuyer sur le *principe de localité* pour le choix des éléments à supprimer, notamment sur la *localité spatiale* ou la *localité temporelle* dont les définitions sont données ci-dessous.

- ✓ La localité spatiale stipule que, plus un élément est proche d'un autre qui vient d'être référencé, plus il a de chance d'être accédé à son tour.
- ✓ Le principe de la localité temporelle stipule qu'un élément récemment accédé a de grandes chances d'être de nouveau référencé.

La politique **aléatoire** (*random*) [11] est une des stratégies les plus simples à développer. Avec cette stratégie, la victime est choisie au hasard. Bien que cette politique soit adaptée à des accès uniformes aux éléments du cache, elle n'est pas vraiment optimale. En effet, elle peut conduire à la suppression d'éléments populaires. Elle constitue néanmoins une référence pour juger d'autres politiques. En effet, une stratégie peut être qualifiée d'« efficace » si elle parvient à utiliser l'information dont elle dispose (moment d'accès, taille des entrées, etc.) de manière à engendrer moins de défauts que la politique aléatoire, qui n'a besoin d'aucune information, ni gestion supplémentaires.

Différentes politiques de remplacement se basent sur le moment d'entrée d'un élément en cache. Ainsi les politiques **FIFO** du premier entré premier sorti et **LIFO** du dernier entré premier sorti choisissent respectivement le premier et le dernier élément ajouté en cache comme victime du remplacement. Ces politiques engendrent peu de surcoût pour la gestion des entrées, puisque la seule information nécessaire est le moment d'entrée. Malheureusement, des éléments populaires peuvent être choisis comme victime. De plus, ces politiques souffrent de *l'anomalie de Belady*. Autrement dit, augmenter les ressources physiques allouées au cache peut accroître le nombre de défauts. Par exemple, avec la séquence d'accès **1 2 3 4 1 2 5 1 2 3 4 5**, un cache de trois éléments utilisant la politique **FIFO** produit neuf défauts, alors que ce nombre s'élève à dix pour un cache de quatre éléments. L'anomalie de Belady dans [21] caractérise un cache dont l'augmentation de l'espace physique ne produit pas une réduction du nombre de défauts .

Certaines politiques de remplacement prennent en compte le moment d'utilisation des entrées présentes en cache. Pour ce faire, le cache garde comme information le moment où un élément a été accédé pour la dernière fois. Il est, par conséquent, nécessaire de modifier cette information à chaque fois qu'un élément est utilisé. Les politiques **LRU** de l'élément le moins récemment utilisé et **MRU** de celui le plus récemment utilisé choisissent l'élément, respectivement le moins et le plus récemment utilisé, comme victime. La première politique est utile pour une forte localité temporelle. La deuxième est plus particulièrement utile pour

les séquences d'accès cycliques, puisqu'un élément qui vient d'être accédé est sans doute celui qui sera le plus tardivement accédé à l'avenir.

Une autre approche possible consiste à remplacer les éléments en fonction de leur fréquence d'utilisation. Le choix de la victime du remplacement se fait alors en fonction du nombre d'accès à un élément depuis que celui-ci est présent en cache. Parmi ces politiques, nous pouvons citer **NFU**, **NRU** ou encore **LFU**. Il est à noter qu'il est parfois nécessaire d'utiliser un autre mécanisme pour différencier deux éléments ayant la même fréquence d'utilisation. Ainsi, **LFUPP** est une variante de LFU, qui intègre une notion de priorité (réinitialisée périodiquement) pour distinguer les éléments de plus faible fréquence.

La taille des entrées peut également être considérée lors du choix des victimes du remplacement. En effet, il est parfois intéressant de favoriser la suppression d'éléments de grande taille, plutôt que de supprimer de nombreux éléments de petite taille pour libérer un espace équivalent. Ainsi, avec **SIZE**, les éléments remplacés correspondent aux entrées du cache les plus volumineuses. Comme pour le remplacement sur la fréquence d'utilisation, un autre mécanisme de décision doit être proposé pour des éléments de taille équivalente (par exemple, différencier les éléments en utilisant **LRU**).

Certaines politiques de remplacement reposent sur un rapport *coût/bénéfice* pour déterminer les éléments à supprimer, comme suggéré dans [21]. Ce rapport peut être calculé de différentes manières. Le **coût** peut ainsi correspondre à la complexité de l'évaluation, comme dans **LNC-R** qui intègre dans le choix de la victime du remplacement le coût de calcul de l'entrée, en plus de la taille et des moments d'utilisation. Ceci permet notamment de privilégier un calcul d'une jointure, plutôt que le résultat d'une projection. Le coût peut aussi prendre en compte le temps de récupération des éléments, comme dans **GDS**. Le **bénéfice** peut, quant à lui, être exprimé en fonction du moment d'utilisation d'une entrée, les éléments les plus récemment utilisés pouvant être considérés comme les plus pertinents à conserver. A noter que parfois le bénéfice est également pris en compte pour les éléments absents du cache, comme dans **LVCT** [14]. Cette valeur peut alors être utilisée par la politique d'admission pour éviter qu'une entrée ne soit ajoutée au détriment d'éléments plus intéressants.

Les différentes politiques de remplacement présentées jusqu'à maintenant sont orthogonales. En conséquence, elles peuvent être combinées, de plusieurs façons, en vue de proposer d'autres stratégies. Il est d'abord possible de choisir la victime en fonction de plusieurs caractéristiques. Par exemple, certains caches Internet choisissent les victimes en

considérant conjointement la fréquence et le moment d'utilisation, à l'instar de **LRFU**. D'autres approches consistent à distinguer différents groupes d'entrées et à appliquer des stratégies différentes pour chacun d'eux. Ceci est notamment le cas dans les systèmes de gestion de bases de données utilisant la politique de remplacement **DBMIN**, où les politiques **MRU** et **LIFO** sont utilisées respectivement pour les séquences cycliques et cycliques hiérarchiques, alors que la politique **LRU** est appliquée pour les séquences n'appartenant à aucun groupe particulier.

3.3. Résolution d'un défaut de cache

Le *protocole de résolution* capture l'ensemble des actions à effectuer afin de résoudre un défaut de cache. En particulier, elle détermine la source de données utilisée, notamment si plusieurs serveurs sont disponibles ou s'il est possible de contacter d'autres caches. D'autres considérations peuvent également être abordées, comme par exemple, la gestion des communications ou encore la gestion de connexion sécurisée (dans ce cas le protocole de résolution peut, par exemple, fournir un identifiant et un mot de passe). Le protocole de résolution définit la procédure exécutée pour récupérer des éléments lorsque des défauts de cache se produisent.

4. Fonctionnalités optionnelles d'un cache

Les fonctionnalités optionnelles d'un cache regroupent des techniques tels que le préchargement, la politique d'admission, la mise en cache négative ou encore l'allocation des ressources. Ces fonctions engendrent des coûts supplémentaires à la gestion du cache et doivent par conséquent être intégrées avec prudence. Cependant, si elles sont judicieusement appliquées, la qualité du service de cache est améliorée.

4.1. Pré chargement

Les algorithmes de pré chargement visent à récupérer des éléments non encore demandés mais susceptibles de l'être dans un avenir proche. La plupart du temps, l'ajout d'un élément en cache se fait à la suite d'une demande d'un utilisateur ou d'une application. Néanmoins, d'autres événements peuvent être à l'origine de cette création. Dans un contexte sans fil, une unité mobile peut profiter d'une connexion à un réseau pour récupérer un ensemble d'éléments qu'elle pourra éventuellement utiliser plus tard en mode déconnecté. Dans une hiérarchie de caches Internet, si le nombre de demandes d'un élément dépasse un certain seuil sur un cache de haut niveau, il est possible de créer des copies sur des caches de niveau inférieur pour répartir la charge (on parle alors de *push caching*).

En plus du moment du pré chargement, il est important de choisir les éléments à précharger. Différentes approches peuvent être envisagées. Les éléments peuvent, par exemple, être pré chargés aléatoirement ou en fonction de statistiques sur les accès passés. Dans certains contextes, la structure des demandes ou des éléments requis peut être utilisée. Ainsi pour une page Internet, les liens hypertextes peuvent être utilisés pour télécharger les pages correspondantes.

4.2. Politique d'insertion et d'admission

Lorsqu'un cache reçoit une demande d'un client, le module de gestion des ressources doit décider s'il peut accepter de servir ce client ou non. La première vérification faite est la présence ou non du document dans la zone de stockage. Si le document n'est présent, la politique d'insertion est interrogée afin de savoir s'il est judicieux de récupérer ce document et de le stocker, tout en le fournissant au client demandeur. Il existe, dans la littérature, de nombreuses politiques d'insertions. Les plus connues étant les politiques *Ignore First Hit* [12] et celle basée sur la fréquence des demandes. La première politique traite le problème des « *one-timer* », l'ensemble important de document qui ne sont demandés qu'une fois et qui polluent les caches, par une méthode simple : on accepte de stocker un document que s'il a déjà été demandé. Un historique des demandes doit donc être maintenu. Les méthodes se basant sur la fréquence des demandes utilisent le même principe, sauf que le nombre de demandes doit être plus important, supérieur à un seuil fixé. Bien entendu, l'espace de stockage disponible doit, dans tous les cas, être suffisant au moment de l'admission.

Lorsque le document est accepté par la politique d'insertion, ou s'il est déjà présent, une autorisation est demandée à la politique d'admission. Le rôle de ce module est de s'assurer que le cache possède des capacités suffisantes pour prendre en charge un nouveau client, sans pénaliser les autres connexions en cours. Les contraintes qui doivent être vérifiées concernent la bande passante disponible sur le réseau, la bande passante au niveau du disque dur, qui est le second goulot d'étranglement, la mémoire libre et le temps processeur. On distingue trois familles de contrôles d'admission [22] :

- ✓ Les contrôles déterministes qui n'autorisent de nouveau client que si les autres connexions ne sont nullement altérées ;
- ✓ Les contrôles statistiques qui autorisent une possible baisse de qualité si la probabilité n'est pas trop forte ;

- ✓ Les contrôles dits « best effort » qui, eux, ne garantissent rien, si ce n'est le fonctionnement du cache en mode dégradé. Ce n'est qu'une fois l'autorisation du contrôleur d'admission obtenue qu'une réponse positive peut être envoyée au client. Dans les autres cas une réponse négative lui est retournée afin qu'il se dirige vers un autre cache ou vers le serveur d'origine

4.3. Cache négatif

Un cache négatif est un dispositif qui informe qu'une demande ne peut temporairement pas être résolue. Il marque temporairement certains éléments comme inaccessibles et ainsi empêche de contacter les serveurs pour les récupérer. Ceci évite alors de surcharger les liens de communication par des demandes envoyées à des serveurs en panne ou coupés du réseau. Dans ce cas, une demande peut malheureusement parfois ne pas être résolue alors que le serveur est de nouveau accessible. C'est pourquoi, il est important de bien configurer la période de validité des éléments du cache négatif.

4.4. Politique d'allocation

La politique d'allocation est en charge de répartir les ressources de stockage proposées par le cache entre les différents utilisateurs de celui-ci.

Une politique d'allocation peut être mise en œuvre afin de distinguer les accès aux éléments du cache. La répartition autorise par exemple à favoriser certains utilisateurs prioritaires ou certaines applications critiques en leur allouant un espace physique plus important. Elle peut également être utilisée conjointement avec une stratégie de préchargement, les éléments préchargés étant alors distingués des éléments réellement accédés. En cas d'accès par un client, un élément préchargé deviendra un élément accédé et sera alors déplacé dans l'espace correspondant. Dans le cas de la politique de remplacement DB-MIN, les éléments du cache sont regroupés selon les séquences d'accès et associés à la stratégie de remplacement correspondante. Ainsi un espace d'accès peut par exemple être réservé pour les séquences d'accès cyclique, le remplacement des entrées pour cet espace se faisant alors à l'aide de la politique MRU.

5. Fonctionnalités gérées par des services externes au cache

Dans cette section, nous présentons les fonctionnalités utilisées par un cache, mais pouvant être à la charge d'un autre service. A la différence des caractéristiques présentées jusqu'à maintenant, ces outils sont partagés entre le cache et d'autres entités du système (des serveurs ou des miroirs par exemple). Deux fonctionnalités sont particulièrement pertinentes

dans les applications (largement) réparties : le contrôle de *concurrency* et la gestion de la *synchronisation*.

5.1. Contrôle de concurrence

Un contrôle de la concurrence est nécessaire si des accès à une même donnée par deux processus distincts peuvent entraîner des dysfonctionnements. Ce genre de situation peut se produire par exemple si une écriture est exécutée au même moment qu'une lecture. Le contrôle de concurrence vise à garantir une exécution correcte, en planifiant les accès de manière à éviter les conflits. La notion la plus reconnue pour assurer l'exécution concurrente des accès correspond à la *sérialisabilité*. Cette notion assure que l'exécution de requêtes (éventuellement entrelacées) conduit au même résultat que ces requêtes exécutées de manière séquentielle, sans pour autant préciser un ordre. Les méthodes de contrôle de concurrence peuvent être rangées dans deux catégories : pessimiste et optimiste. Avec une approche **pessimiste**, un client doit d'abord demander la permission d'accéder à un élément, seuls les accès non conflictuels étant alors autorisés. Une approche **optimiste**, quant à elle, autorise les accès sans demande de permission, les conflits étant résolus au moment de la validation.

5.2. Synchronisation

Un mécanisme de synchronisation a pour but d'assurer le degré de cohérence souhaité entre les copies d'un même élément. Ceci est nécessaire pour des caches ayant un droit de modification de leur contenu ou pour lesquels les éléments stockés peuvent être modifiés sur les serveurs.

La synchronisation des copies peut être exécutée à différents moments. En reprenant la classification proposée dans [22], les synchronisations peuvent être regroupées selon les événements qui engendrent le processus. La *synchronisation temporelle* peut exprimer une durée maximale attendue avant la propagation d'une mise à jour, ou une périodicité spécifiant la fréquence de la synchronisation. La synchronisation sur la valeur peut spécifier un nombre de modifications pouvant être appliquées avant qu'une copie ne soit mise à jour, ou encore pour les objets numériques, déclencher la synchronisation sur la différence de valeur entre les copies dépasse un certain seuil. Des événements particuliers peuvent également être considérés comme, par exemple dans des environnements mobiles, la connexion d'une station mobile sur un réseau filaire.

L'initiative de la synchronisation peut se faire de deux manières : les synchronisations de type *pousser* (*push*) et *tirer* (*pull*). Avec une synchronisation de type *pousser*, comme par

exemple le mécanisme de *callback* du système de fichiers *Andrew*, les mises à jour sont diffusées de la copie modifiée sur le serveur vers les copies stockées en cache. Dans un système basé sur une synchronisation de type **tirer**, à l'instar du système de gestion de fichiers **NFS**, chaque accès à un élément du cache engendre une consultation de la copie présente sur le serveur pour vérifier si cette dernière a été modifiée. La synchronisation peut alors dépendre de l'âge des éléments du cache, comme dans le système de fichiers *Alex*, ou reposer sur une durée de vie ou **TTL**, même si celle-ci est difficile à déterminer, son estimation pouvant ainsi être laissée à la charge du client lui-même.

La nature des mises à jour désigne le type d'information échangée lors du processus de synchronisation. Deux approches peuvent être utilisées : les protocoles à diffusion des écritures et les protocoles à invalidation. Alors que le premier transfert des valeurs, des différences de valeurs ou la procédure exécutée, le second peut marquer à tout moment une copie comme non synchronisée.

La synchronisation peut être exécutée selon différentes topographies, qui dépendent de la priorité de certains caches, ou encore de la topologie du réseau. Ainsi, sur un anneau, la mise en cohérence se fait entre deux copies, alors que sur une clique, une approche envisageable consiste à diffuser les mises à jour à toutes les copies. Il est également possible de diffuser à un sous-ensemble, notamment dans une approche hiérarchique, où la diffusion peut se faire uniquement sur les caches fils et non sur toute la descendance (chaque cache fils pouvant ensuite continuer la propagation).

Finalement, la synchronisation doit considérer la capture des mises à jour. Un tel mécanisme peut être assuré à l'aide d'un support particulier, comme un journal (enregistrant les requêtes effectuées sur une copie) ou une copie ombre (copie de copie gardant une trace des valeurs d'un élément avant certaines modifications). Il est également possible pour capturer les mises à jour d'avoir recours à des mécanismes déclencheurs, comme par exemple des appels de méthode, ou simplement en consultant la copie source.

II. Caches coopératifs

La section I a présenté les caractéristiques générales des caches. Cette section a pour objectif de présenter plus en détail les caches coopératifs, visant à optimiser respectivement l'utilisation des ressources globales. Elle s'intéresse aux techniques de coopérations entre caches.

Le World Wide Web est devenu une ressource fondamentale pour la distribution des informations. Cela a conduit à un accroissement immense de la quantité d'informations disponibles ainsi que le nombre de données requêtées. Les proxies caches ont été longtemps utilisés dans le but d'améliorer les performances pour une communauté d'utilisateurs [13]. Cependant, et du à la nature distribuée du Web et les garanties offertes aux utilisateurs, plusieurs systèmes ont été proposés afin d'une part de rendre le Web plus attractif en réduisant le temps de latence, et d'autre part de réduire la bande passante utilisée. Au lieu de laisser chaque cache opérer séparément du reste du réseau, il est souvent bénéfique de permettre aux proxies caches d'utiliser les ressources des autres caches. Cette technique permet de réduire le temps de latence, la charge du serveur et le trafic réseau. On appelle un tel système, *système de caches coopératifs*.

1. Les systèmes de caches coopératifs :

L'augmentation de la quantité d'informations requêtées dans le Web a engendré un besoin de développer des mécanismes qui aident à la gestion de cette utilisation croissante du Web. Une des solutions proposées est l'utilisation des caches coopératifs. Dans cette approche, plusieurs proxies cache distribués peuvent coopérer pour partager leurs ressources [9].

Les caches coopératifs forment un groupe de serveurs proxies HTTP qui partagent leurs objets cachés. Bien que plusieurs recherches [9] aient montré l'intérêt des proxies cache en terme de performances, en réduisant le temps de latence, la charge du serveur, et l'ensemble du trafic réseau. Le débat n'est pas clos sur les politiques de coopération entre proxies cache. Nous trouvons dans la littérature trois grands systèmes de caches coopératifs. Ces systèmes sont « Geographical Push » [14], « Hierarchical » [15], et « Adaptive » [19].

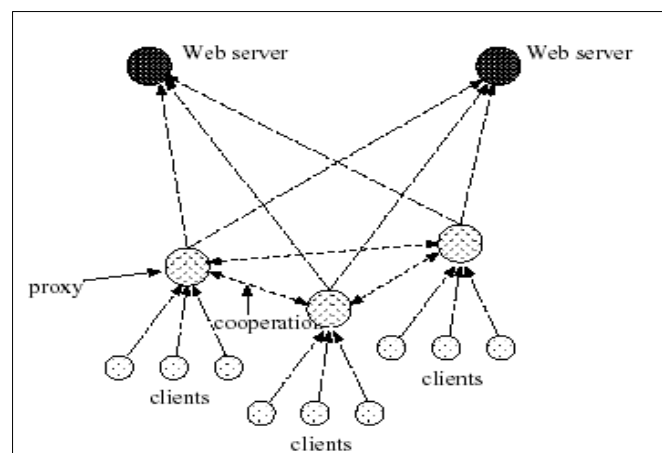


Figure 2.2. Schéma de proxies caches coopératifs.

1.1. Geographical Push caching :

Ce système a été proposé par Gwertzman et al [14]. L'idée de base est de garder les données proches des clients qui les demandent. Dans une telle architecture, le serveur primaire est responsable de ce qui doit être caché, où et quand. Il garde une trace de la fréquence et provenance de toutes les requêtes. Quand le nombre de requête atteint un certain seuil, le serveur primaire envoie (Push) le document demandé vers le cache distant. Le choix du cache approprié se fait en se basant sur l'espace disponible, la charge et l'origine des requêtes.

Pour récupérer un document, le client s'adresse directement au serveur. Ce dernier, au lieu de servir directement la requête, indique au client la position du cache le plus proche où il peut trouver le document. Le serveur peut toutefois traiter la requête lui-même, s'il est le plus proche.

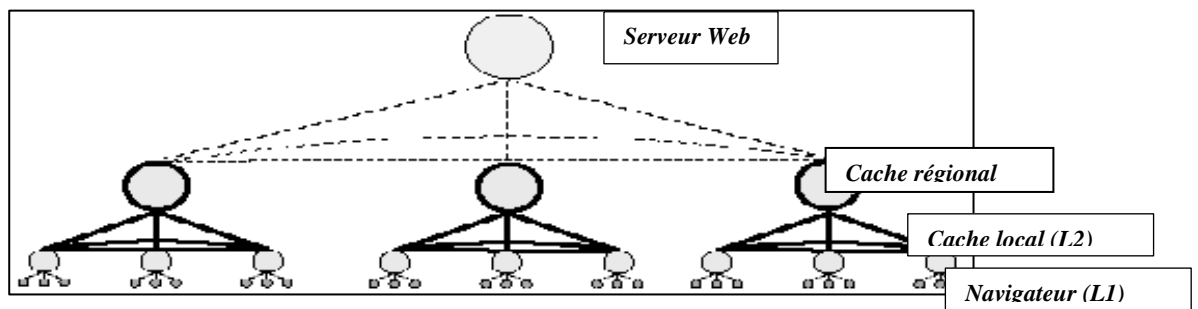


Figure 2.3. Structure hiérarchique de proxy caches coopératives.

(Lignes en gras indique la communication des caches L3).

Dans un système hiérarchique, quand une requête provenant d'un utilisateur, ne peut être satisfaite par le cache du client (L1), elle est redirigée vers les caches institutionnels (L2), ce dernier envoie les requêtes qu'il ne peut pas satisfaire aux caches régionaux (L3). A ce niveau le cache contacte directement le serveur d'origine. Quand un document est trouvé, il traverse la hiérarchie, en laissant une copie dans chaque cache intermédiaire. Placer des copies redondantes des documents dans les niveaux intermédiaires réduit le temps de connexion [20]. Ainsi, les prochaines requêtes pour le même document montent dans la hiérarchie jusqu'à ce que la requête trouve le document. L'environnement logiciel le plus utilisé pour les caches hiérarchiques est Squid, qui est un descendant du projet Harvest [15].

La structure de caches hiérarchiques est efficace pour une consommation modérée de la bande passante, particulièrement quand les caches coopératives n'ont pas une grande vitesse de connectivité. Cependant, malgré leurs bénéfices, il existe plusieurs problèmes liés aux caches hiérarchiques [20] :

1. Pour mettre en place une hiérarchie, les serveurs cache ont souvent besoin d'être placé à des points d'accès stratégiques du réseau. Cela demande une grande coordination entre les serveurs caches participants.
2. Chaque niveau de la hiérarchie introduit un délai additionnel.
3. Le niveau le plus haut de la hiérarchie peut représenter un goulot d'étranglement, et par conséquent il introduit des délais dans les réponses.
4. Des copies multiples du même document sont sauvegardées à différents niveaux du cache.

1.2. Adaptive caching :

Adaptive caching [13] explore le problème du cache comme moyen d'optimiser la diffusion des données. La première idée d'Adaptive caching a été proposée par L. Zhang et al [19]. Dans cette approche, des groupes de caches forment des mailles (Mesh), qui se chevauchent. Les groupes de caches sont auto configurables, et s'adaptent aux changements de réseaux.

Adaptive caching [13] emploie le Cache Group Management Protocol (CGMP). CGMP spécifie comment les mailles sont formées, et comment les caches rejoignent et quittent ces mailles. En général, les caches sont organisés en groupes multicast chevauchés, et utilisent un vote et la technique de feedback pour estimer le gain avant d'admettre ou d'exclure un membre du groupe.

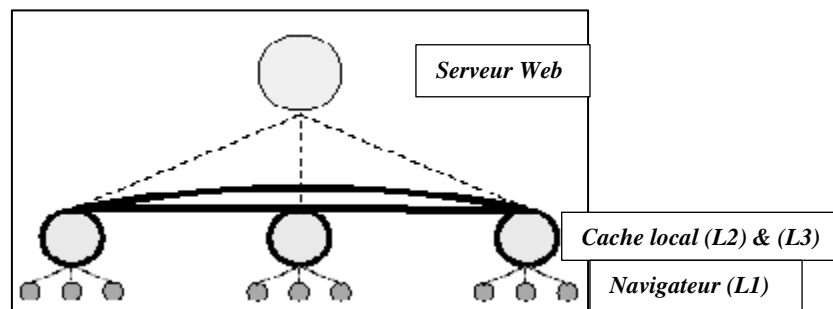


Figure 2.4. Structure en maille de proxy caches coopératives

(Lignes en gras indique la communication des caches L3).

Chacun des systèmes de caches coopératifs cités auparavant est basé sur un protocole qui permet de communiquer avec les autres caches de la communauté de coopération. Ce protocole définit aussi la manière de rechercher les objets dans les caches voisins. La section suivante s'intéressera aux protocoles de communication inter-cache. Nous décrivons le principe de fonctionnement de chacun d'eux.

1.3 Protocoles de Communication inter-caches :

Les systèmes de caches tendent à être composés de multiples caches distribués pour améliorer l'extensibilité et la disponibilité du système. Afin de permettre une telle fonctionnalité un protocole de communication entre les caches coopératifs est nécessaire. On trouve dans la littérature plusieurs systèmes de communication inter-cache. Nous détaillerons juste après les plus importants :

1.3.1. Internet Cache Protocol (ICP) :

Les chercheurs du projet Harvest [15], ont développé un protocole de communication entre les caches appelé ICP (Internet Cache Protocol) [16]. Les groupes de caches peuvent bénéficier du partage des données entre eux, de la même manière qu'un groupe d'utilisateur partage un cache. ICP fournit des méthodes rapides et efficaces pour la communication inter-caches, et il offre des mécanismes pour établir des caches hiérarchiques complexes [16].

ICP est principalement utilisé dans une maille de cache, pour trouver des objets Web spécifiques dans le voisinage d'un cache. Il donne des indications sur l'emplacement des objets Web. A la demande d'un objet qui n'est pas caché localement, le serveur proxy diffuse (multicast) des requêtes ICP (ICP_QUERY) sur tous ses voisins (avec lesquels il est configuré à fonctionner). Les voisins renvoient des réponses ICP indiquant un « Hit » ICP_HIT1 (le voisin possède l'objet demandé) ou un « Miss » ICP_MISS2 (le voisin ne possède pas l'objet demandé).

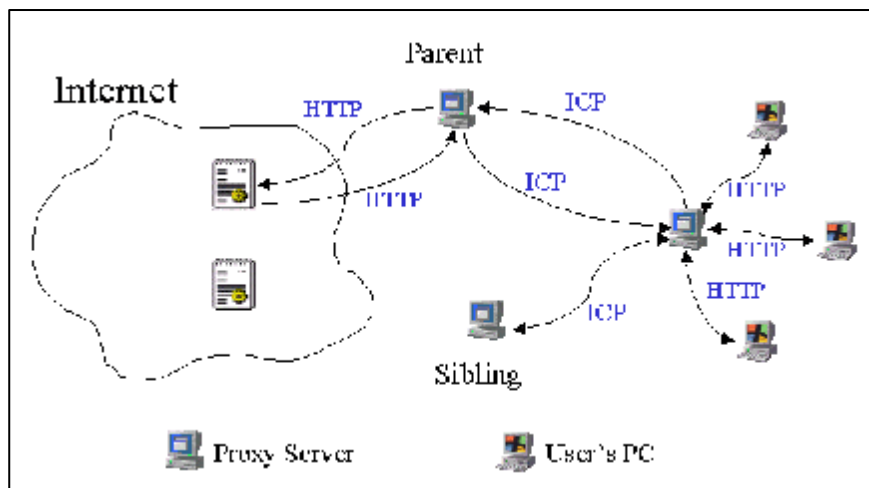


Figure 2.5. Le protocole de communication inter caches ICP.

Un message ICP est constitué d'un entête de taille fixe (20 octets) et de l'URL [16]. Dans la pratique ICP est implémenté au-dessus d'UDP, pour accélérer les messages requête/réponse

échangés. Il ne contient aucune information des entêtes http associés à un objet. Les entêtes HTTP doivent contenir des contrôles d'accès et des directives de cache.

En dépit de son succès et son utilisation répandue, ICP n'est pas vraiment un protocole très extensible. Dès que le nombre de proxies coopératifs augmente, le nombre de message émit pour retrouver un objet devient rapidement prohibitif. Une étude effectuée par Cao et al [9] a montré que les messages ICP augmentent le trafic UDP par un facteur de 73 à 90%.

1.3.2. Cache digests : (Résumés des caches)

Cache digest est une technique d'optimisation pour les caches coopératifs. Elle permet aux proxies de diffuser des informations sur leurs contenus disponibles aux voisins dans une forme compacte. Ce protocole est proposé par A. Rousskov et al [17], comme une alternative d'ICP.

Cache digest contient, en format compressé, une information indiquant si une URL particulière est dans le cache ou non. La forme compacte de toutes les clés du cache est obtenue en utilisant la technique de *bloom filter* [9]. Avec cette approche, les serveurs de cache s'échangent leurs résumés périodiquement.

Quand un cache reçoit une requête pour un objet (URL) de son client, il peut utiliser les résumés de ces voisins, pour trouver celui qui détient cet objet.

Pour réduire la taille du cache digest, les auteurs [17] ont proposé d'utiliser une partie du codage MD5 (8 bit seulement à la place 128 bits) construit à partir de l'URL, et quatre fonctions de hachage pour le positionnement des bits à 1. Un autre point important est à signaler : chaque cache détermine la taille de son résumé indépendamment de ses voisins.

Cache digest a été implémenté au-dessus de Squid 1.2. Rousskov et al [17] ont fait des comparaisons entre Cache digest et la version Squid (à base de ICP). Ils ont trouvé que le temps de service moyen est amélioré de 100 millisecondes et plus, quand ils utilisent Cache digest à la place de ICP. De plus, avec des paramètres adéquats (période de mise à jour), cette technique consomme moins de bande passante. Cependant, tandis que ICP génère un flux stable de petits paquets, le volume de transfère de Cache digest est un peu élevé. En conclusion, cette approche n'est pas vraiment convenable sous des réseaux à faible débit, ou congestionné.

1.3.3. Summary cache :

Dans la même période où Rousskov et al [17] étaient entrain de développer Cache digest, P. Cao et al [9] de l'université de Wisconsin, Madison étaient en train de développer le Summary cache. Cette approche est très similaire à Cache digest.

Dans Summary cache, chaque proxy garde un résumé compact des répertoires de cache, de tous les autres caches coopératifs. Quand un *MISS*³ survient, le proxy enquête tous les résumés de ces voisins (localement), pour voir si la requête peut donner un *HIT*⁴ dans d'autres proxies voisins. Si un tel cas se présente, le proxy envoie des messages de requête seulement aux proxies qui paraissent prometteurs d'après leurs résumés. Comme Cache digest, Summary cache n'est pas exact à tout moment (faux HIT vs faux MISS). Un *faux HIT*⁵ se produit quand le résumé indique un HIT, alors que l'objet n'est plus dans le cache.

Dans le cas d'un faux MISS, le document n'est pas caché dans certains caches, mais leur résumé indique qu'il l'est. Ces erreurs affectent le *Hit ratio*⁶ total et le trafic inter-proxies, mais pas la validité du système de caching [9]. Les résumés des répertoires sont stockés en utilisant la technique du bloom filter [9,17].

Les simulations effectuées par P. Cao et al [9], ont montré que le protocole *Summary cache* réduit les messages inter-proxies par un facteur de 25 à 60% ; réduit la consommation de la bande passante de plus de 50%, et élimine entre 30% à 95% de la charge CPU, par rapport à Squid. Summary cache a été implémenté pour accroître ICP dans Squid 1.1.13.

A première vue, on ne remarque pas une grande différence entre Cache digest et Summary cache. En réalité la différence majeure entre eux est que la mise à jour des résumés (summaries) s'effectue ici en utilisant la technologie Push (c'est à dire le serveur doit maintenir l'état de tous ses fils). C'est d'ailleurs l'inconvénient de cette approche, alors que pour le Cache digest, il y a un échange périodique des résumés.

1.3.4. Mécanismes de validation/réplication de documents :

Les caches proxies coopératifs sont des groupes de caches qui partagent des objets cachés et coopèrent entre eux, pour faire le même travail comme un seul cache web. Cependant, le cache a le désavantage de retourner des documents non frais. Cela signifie que le document n'est pas consistant avec la version qui est sur le serveur. Ce problème peut être détourné en utilisant les mécanismes de validation/réplication.

La validation/réplication, est un mécanisme qui permet de distribuer et de maintenir la consistance des documents, c'est un point très important pour l'implémentation d'une architecture de distribution de documents. On trouve deux grandes stratégies pour des documents dans un système de cache [18], le mécanisme *pulling* et le mécanisme *pushing*.

Dans le mécanisme *pulling*, chaque cache interroge périodiquement le serveur origine (ou bien le cache du niveau supérieur selon l'architecture définie). Dans le mécanisme de *pushing*, les caches n'ont pas besoin d'interroger le serveur périodiquement pour la fraîcheur

du document, ce dernier distribue les documents modifiés aux caches intéressés ou abonnés. A l'intérieur de ces stratégies, on peut définir plusieurs mécanismes de validation/réplication, qu'on regroupe en trois blocs :

TTL, Validate Verification et le *Callback* (invalidation).

1)- *TTL*: les serveurs ajoutent une marque de temps (time stamp) pour chaque document requêté par le cache. Cette marque définit la période de temps durant laquelle le document est encore valide. Par exemple le champ *Expire* du protocole HTTP/1.0 est un bon exemple de cette approche.

2)-*Validate Verification* : ce mécanisme est lié à TTL. Le cache reçoit un document avec une marque de temps qui indique la date de la dernière modification de ce dernier. Quand un client demande ce document, le cache envoie un message de vérification au serveur, en incluant la marque de temps enregistrée dans le document.

Le serveur valide la marque de temps de ce document avec la date de dernière modification du document, et envoie un message de « No-Modification » (le document n'a pas été modifié), ou bien le nouveau document avec la nouvelle marque de temps.

Cette approche est implémentée dans HTTP/1.0 en utilisant la balise If-Modifie-Since (IMS).

3)-*Callback (invalidation)* : chaque serveur garde une trace de tous les caches qui ont demandé une page donnée, et quand cette page change, il avertit ses caches. Cette approche présente une faible extension, vu qu'elle nécessite la sauvegarde de la liste de tous les demandeurs du document, ainsi qu'une augmentation de la charge du système et du réseau, induite par l'obligation de contacter tous les demandeurs d'un document à chaque modification. Cependant, ce problème d'extensibilité peut être surmonté en utilisant la diffusion (multicast) pour la transmission de l'invalidation (ICM), en attribuant un groupe multicast pour chaque page. La diffusion résout le problème d'extensibilité au niveau du serveur, mais il en crée d'autre au niveau des routeurs [18].

1.3.5. Performances des systèmes de caches coopératifs :

Les systèmes de caches coopératifs ont prouvé leurs performances dans plusieurs domaines : système de fichier, mémoire virtuelle, et dans des réseaux locaux.

D'après des études effectuées par Wolman et les autres, les auteurs se sont posés la question suivante : *est ce que n'importe quelle forme de coopération de cache fournit un bénéfice significatif au-delà de deux niveaux ?*

Ils ont montré que le bénéfice de la coopération de cache est dû simplement à l'augmentation du nombre d'utilisateurs partageant un ensemble de documents Web.

De plus, la performance s'améliore seulement au-delà d'une certaine taille de population. Les auteurs ont montré aussi qu'une grande communauté de proxies gagne peut en coopérant entre eux, et qu'il est plus intéressant d'avoir des petites communautés qui coopèrent à petite échelle. En conséquence, ils ont trouvé qu'il n'y pas de besoin à une large coopération de proxies. Cette étude s'est adressée à la question fondamentale dans le domaine des caches coopératifs : quelle est l'étendue et la nature du partage des documents sur le Web?

Nous parlerons dans les sections suivantes la notion d'agent et système multi-agent.

III. La notion d'agent

Depuis quelques années, le terme "agent" est de plus en plus utilisé aussi bien dans le monde de la technologie de l'information que dans le monde des télécommunications. On trouve, cependant, plusieurs définitions concernant l'agent qui se ressemblent mais diffèrent suivant le type d'application pour lequel l'agent est conçu.

1. Définition: dans [25]

« Un agent est une entité logicielle ou matérielle, à qui est attribuée une certaine mission qu'elle est capable d'accomplir de manière autonome, disposant d'une connaissance partielle de ce qui l'entoure (son environnement), et agissant par délégation pour le compte d'une personne ou d'une organisation ».

Cette définition assez générale insiste sur deux caractéristiques importantes:

1) l'agent est une entité qui agit par *délégation*, l'agent doit respecter la stratégie du client vis à vis des choix qu'il est amené à faire, cela afin que le client soit responsable des tâches effectuées par l'agent.

2) l'agent est une entité *autonome* qui dispose de son propre environnement, c'est à dire que l'agent n'est pas lié constamment au client qui l'utilise.

Nous trouvons dans la littérature une multitude de définitions d'agents qui se ressemblent toutes, mais se diffèrent selon le type d'application pour laquelle est conçu l'agent.

2. Les différentes multitudes de définitions d'agents

- ✓ **Agent matériel:** c'est un agent doté d'un équipement physique, tel qu'un robot.
- ✓ **Agent logiciel:** Un agent logiciel est purement logique, incluant du code, des données, et un état.

✓ **Agent cognitif** : Un agent est cognitif ou encore intentionnel, s'il ne se contente pas d'agir en fonction des conditions de son environnement mais en fonction de ses buts propres et s'il possède une intentionnalité, c'est-à-dire une volonté consciente d'effectuer un acte.

✓ **Agent mobile**: C'est un agent matériel capable de se déplacer dans l'environnement ou bien un logiciel pouvant migrer d'un site à un autre en cours d'exécution en fonction de ses besoins et qui suit parfois un itinéraire.

✓ **Agent adaptable**: un agent est dit *adaptable* si certains de ses mécanismes internes, opérationnels (envoi de messages, déplacement...) ou fonctionnels (comportement), sont modifiables en cours d'exécution.

✓ **Agent social**: ce type d'agents opère des interactions évoluées avec d'autres agents au sein de son environnement.

Notons que ces différents types possèdent plus ou moins de recouvrements entre eux. Nous constatons également qu'ils cadrent plus ou moins la définition que nous avons donnée plus haut, car cela dépend des caractéristiques mises en avant selon le domaine en question. En fait, il n'existe pas de compétition entre les différents paradigmes, mais plus tôt, des opportunités de mise en œuvre complémentaire. Le choix dépend de l'architecture et du but poursuivi.

3. Les différentes catégories d'agents [25]

Les agents peuvent être classés en trois catégories relatives à leur réactivité :

✓ **Cognitifs** : Ils peuvent anticiper, prévoir le futur, mémoriser des choses, ils réfléchissent.

✓ **Réactifs** : Ils réagissent directement à l'environnement perçu, par pulsion (ex : les fourmis).

✓ **Hybride** : un mélange des deux. Il y a donc une infinité de systèmes multi-agents différents. En plus de cette différence entre les agents, de leur niveau d'organisation, s'ajoutent les moyens de communication (d'agent à l'agent, d'agent à agent passant par le SMA, par l'environnement (les signaux)...), les problèmes d'interaction et de coopération qui peuvent être réglés de différentes manières, et les actions et comportement qui sont propres à chaque agent.

Remarque :

Pour être opérationnel, l'agent a besoin d'interagir avec l'hôte sur lequel il s'exécute, ainsi que les autres agents.

4. Différence entre objet et agent

On parle d'objet tantôt agent les concepts objet et agent distribué sont très proches mais il n'en est pas de même de l'agent comme entité intentionnelle. En effet les objets n'ont pas but de recherche de satisfaction et le mécanisme d'envoi de messages se résume à un simple appel de procédure. Il n'y a pas de langage de communication à proprement parler. Les mécanismes d'interaction sont donc à la charge du programmeur.

La différence essentielle entre un objet et un agent est qu'un objet est défini par l'ensemble des services qu'il offre (ses méthodes) et qu'il ne peut refuser d'exécuter si un autre objet le lui demande. Par contre, les agents disposent d'objectifs leur donnant une autonomie de décision vis à vis des messages qu'ils reçoivent. Par ailleurs, ils établissent des interactions complexes faisant intervenir des communications de haut niveau.

Du fait, un agent peut être considéré comme un objet doté de capacités supplémentaires : recherches de satisfactions (intentions, pulsions) d'une part, et d'autre part la communication à base de langages plus évolués (actes de langages pour les agents cognitifs, propagation de stimuli pour des agents réactifs). Inversement, un objet peut être considéré comme un agent "dégénéré" devenu un simple exécutant, tout message étant considéré comme une requête. [26]

IV. Système multi-agent (SMA)

« Un système multi-agent est un ensemble organisé d'agents ». Nous ne faisons que suivre ici la définition usuelle du terme système : (un ensemble organisé d'éléments). [27]

1. SMA est un système composé des éléments suivants

- un environnement E , disposant d'une métrique en général,
- un ensemble d'objets O , auxquels on peut associer une position dans E à un moment donné. Ces objets (hormis les agents) sont passifs : les agents peuvent les percevoir, les créer, les détruire et les modifier.
- un ensemble A d'agents, lesquels représentent les entités actives du système,
- un ensemble de relations R qui unissent les objets (et agents) entre eux,
- un ensemble d'opérateurs Op permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O , des opérateurs chargés de représenter l'application de ces opérations et la réaction du monde à cette tentative de modification, que l'on appellera les lois de l'univers.

2. Architecture du SMA

L'architecture du SMA sera liée à la façon dont seront implémentées les communications. On parle ici de SMA en tant que système multi-agents global, c'est à dire, l'infrastructure qui permet aux agents de fonctionner. Il faudrait peut-être parler de SEMA pour Système d'Exploitation Multi-Agents. En effet, le système doit gérer aussi bien l'ensemble des transactions sur le réseau que la gestion des messages ou l'accès aux données. Nous utilisons donc abusivement le terme SMA pour désigner les lieux (ou sites) où s'exécutent les agents, ainsi que l'ensemble des opérations gérées par ces lieux d'exécution.

Conclusion

L'utilisation de caches est fréquente dans de nombreuses applications informatiques. Cependant, la configuration d'une solution, en fonction de son environnement d'exécution, étant cruciale pour obtenir de meilleures performances, les techniques utilisées diffèrent d'une application à une autre.

Afin de proposer une telle solution, il est essentiel de ressortir les caractéristiques remarquables d'un cache, pouvant être vues comme des points d'adaptabilité, et pour chacune d'elles établir un ensemble de valeurs possibles. Ce chapitre présente une proposition de la classification, qui est la suivante : déploiement du cache, fonctionnalités élémentaires, fonctionnalités optionnelles et fonctionnalités gérées par des services extérieurs.

Dans la deuxième partie, nous avons dressé un état de l'art des caches coopératifs. Comme les services Web sont devenus de plus en plus populaires, les utilisateurs souffrent des problèmes de surcharge des serveurs et de congestion du réseau.

Plusieurs recherches ont été faites pour améliorer la performance du Web. Le cache est reconnue comme étant une technique efficace pour diminuer la congestion des serveurs, et de réduire le trafic réseau, et de cette façon, minimiser le temps de latence des utilisateurs.

Exploiter la coopération entre les caches permet d'accroître les performances du système de cache, plutôt que d'avoir plusieurs proxies qui accèdent aux mêmes ressources de manière indépendante. La clé d'efficacité de n'importe quel schéma de coopération réside dans le protocole de communication de cache.

Le prochain chapitre parlera de la conception et de l'implémentation de notre système.

Chapitre 3 : Conception et implémentation

Introduction

L'utilisation du réseau est maintenant devenue chose courante pour une majorité de personnes dans les pays développés. De plus, rien ne semble indiquer une diminution dans le nombre d'utilisateurs puisque la croissance industrielle des pays qualifiés de plus pauvres permet à un nombre toujours plus grand de personnes d'avoir accès à cette technologie. Cependant, à mesure que les gens se familiarisent avec ce nouveau moyen de communication, leurs exigences et leurs attentes deviennent plus difficiles à combler en particulier en ce qui concerne le temps d'accès. Il devient donc primordial de trouver une façon d'accélérer le transfert des données des différents serveurs jusqu'aux clients pour éviter des temps d'attente inacceptables.

Notre travail se porte sur la gestion des caches coopératives pour des données multimédias dans les réseaux mobiles. L'objectif de ce travail est de profiter des avantages des mémoires caches coopératives et distribuées afin d'améliorer les performances des applications Multimédia Interactives et distribuées (AMID). Pour pouvoir réaliser un tel objectif, l'utilisation d'un mécanisme de recherche est indispensable. Pour cela nous avons utilisé deux mécanismes, dont le premier mécanisme est le Tabou et le second est le séquentiel.

A cet effet, comme il y a eu la naissance d'un grand nombre de nouvelles applications qui se développent autour de ce type de réseau dont ces applications sont formées par des entités réparties et la bonne fonctionnalité nécessite la communication et les échanges entre ces entités. Aujourd'hui, le modèle "client/serveur" où les échanges se font par des appels distants à travers le réseau est le modèle le plus utilisé. Dans ce modèle, seul le client représente une application au sens propre du terme et le rôle du serveur est de répondre aux demandes des clients. Le serveur construit ses réponses indépendamment du client, ainsi une partie des données envoyées est inutile augmentant ainsi le trafic sur le réseau.

Le présent chapitre est structuré comme suit : dans la première section nous allons présenter brièvement les outils de développement, en suite dans la deuxième section nous parlerons la base de données de notre système recherche, et la simulation de notre travail. Enfin, une conclusion fait un bilan de cette recherche et présente les perspectives envisagées.

I. Les outils de développement

L'implémentation d'un tel système a besoin d'un langage de programmation et une plateforme multi-agent agents. Donc après s'être informé sur les trois langages proposés, nos choix se sont portés sur JAVA et JADE.

1. Le langage Java

Le langage Java est aujourd'hui largement connu et plébiscité par la communauté informatique. Le but ici n'est donc pas de revenir sur ce qu'est Java en en faisant une présentation générale, mais plutôt de voir en quoi il est particulièrement adapté aux développements de systèmes multi-agents.

En premier lieu, Java est un langage objet. Les agents pouvant être considérés comme des objets avancés, ce type de langage est le mieux adapté pour les développer. De plus, la modularité apportée par les langages objet devient rapidement indispensable dès qu'on modélise des agents dont l'architecture étant peu complexe. Par ailleurs, un des principes fondamentaux des systèmes multi-agents est l'hétérogénéité des architectures et des technologies pouvant être employées pour le développement et l'exécution des agents. Java est un langage semi-interprété qui est exécuté par une machine virtuelle (la JVM déclinée sur de nombreuses plates-formes (Linux, AIX, Mac OS, Solaris, Windows, . . .)). La portabilité du code que cela apporte permet ainsi de déployer un système multi-agents à travers un réseau hétérogène de machines.

D'autre part, Java est un langage construit pour le Web. Un des premiers objectifs était de rendre disponible via le mécanisme d'*applet* un applicatif lourd via le client léger qu'est le navigateur Web. On trouve ainsi le couple de protocoles IIOP et RMI, basés sur la norme CORBA82, permettant l'invocation de méthodes à distance. Ce système est utilisé dans de nombreux outils multi-agents tel que JADE que nous présentons ci-dessous.

2. La Plateforme JADE [28]

JADE (Java Agent Development Framework) est une plate-forme multi-agents développée en Java par CSELT (Groupe de recherche de Gruppo Telecom, Italie), il fournit un environnement de développement et d'exécution des systèmes multi-agents compatibles avec les standards FIPA [28]. JADE comprend deux composantes de base : la plate-forme d'agents compatibles FIPA et un package logiciel pour le développement des agents dans Java. Il fournit les facilités suivantes :

Une plate-forme agent distribuée : la plate-forme peut être distribuée (partagée) entre plusieurs hôtes (hosts) connectés via RMI de Java, de telle façon qu'une seule application

Java, et par conséquence une seule « Machine Virtuelle Java » est exécutée sur chaque hôte.

Un certain nombre de DF (Facilitateurs d'Annuaire) compatibles FIPA : qui peut être activé quand on lance la plate-forme pour exécuter les applications multi-domaines.

Une interface utilisateur graphique (GUI) : pour gérer plusieurs agents et conteneurs d'agents d'un hôte éloigné.

Outils de Débogage : pour faciliter la mise au point d'applications.

Un support d'exécution : pour les activités multiples, parallèles et concourantes des agents via le modèle de comportement (behaviour)

Un transport efficace des messages ACL : à l'intérieur de la même plate-forme. Cette conversion est transparente aux programmeurs intéressés uniquement aux objets Java.

Une bibliothèque de protocoles : compatible aux standards FIPA et prête à être employé pour régir l'interaction inter-agent.

2.1. Architecture logicielle de la plateforme JADE

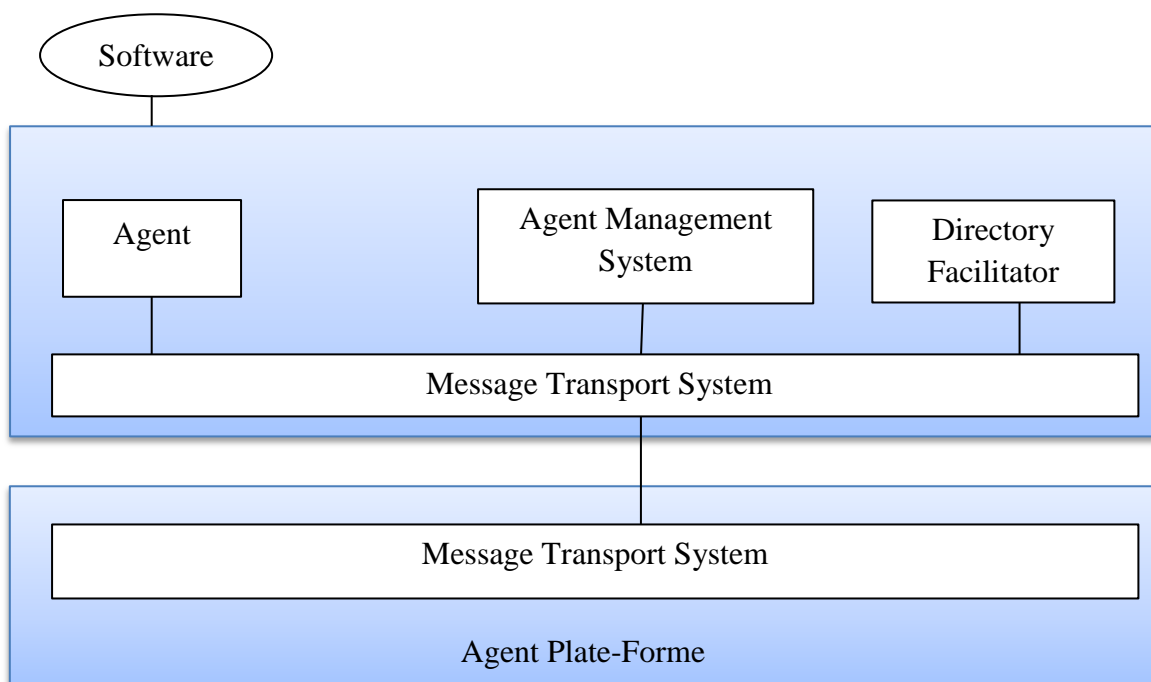


Figure. 3.1 Architecture logicielle de la plate-forme JADE [28]

A travers le schéma donné à la figure 3.1, trois rôles principaux sont dégagés :

Le **Système de Gestion d'Agents** (Agent Management System - AMS) est l'agent qui exerce le contrôle de supervision sur l'accès et l'usage de la plate-forme; il est responsable de l'authentification des agents résidents et du contrôle d'enregistrements.

Le **Canal de Communication entre Agents** (Agent Communication Channel - ACC) est l'agent qui fournit la route pour les interactions de base entre les agents à l'intérieur et en dehors de la plate-forme; c'est la méthode de communication implicite qui offre un service fiable et précis pour le routage des messages.

Le **Facilitateur d'Annuaire** (Directory Facilitator - DF) est l'agent qui fournit un service de pages jaunes à la plate-forme multi-agents. Il faut remarquer qu'il n'y a aucune restriction sur la technologie utilisée pour l'implémentation de la plate-forme: E-mail, basé sur CORBA, applications multithreads Java, etc.

Le standard spécifie aussi le **Langage de Communication d'Agents** (Agent Communication Language - ACL).

La communication des agents est basée sur l'envoi de messages. Le langage FIPA ACL est le langage standard des messages et impose le codage, la sémantique et la pragmatique des messages. La norme n'impose pas de mécanisme spécifique pour le transport interne de messages. Plutôt, puisque les agents différents pourraient s'exécuter sur des plateformes différentes et utiliser des technologies différentes d'interconnexion, FIPA spécifie que les messages transportés entre les plates-formes devraient être codés sous forme textuelle. On suppose que l'agent est en mesure de transmettre cette forme textuelle. La norme FIPA préconise des formes communes pour les conversations entre agents par la spécification de protocoles d'interaction, qui incluent des protocoles simples de type requête-réponse.

2.2. L'interface de la plateforme JADE

Le RMA (Remote Management Agent) permet de contrôler le cycle de vie de la plateforme et tous les agents la composant. L'architecture répartie de JADE permet le contrôle à distance d'une autre plateforme. Plusieurs RMA peuvent être lancés sur la même plateforme du moment qu'ils ont des noms distincts (voir figure 3.2).

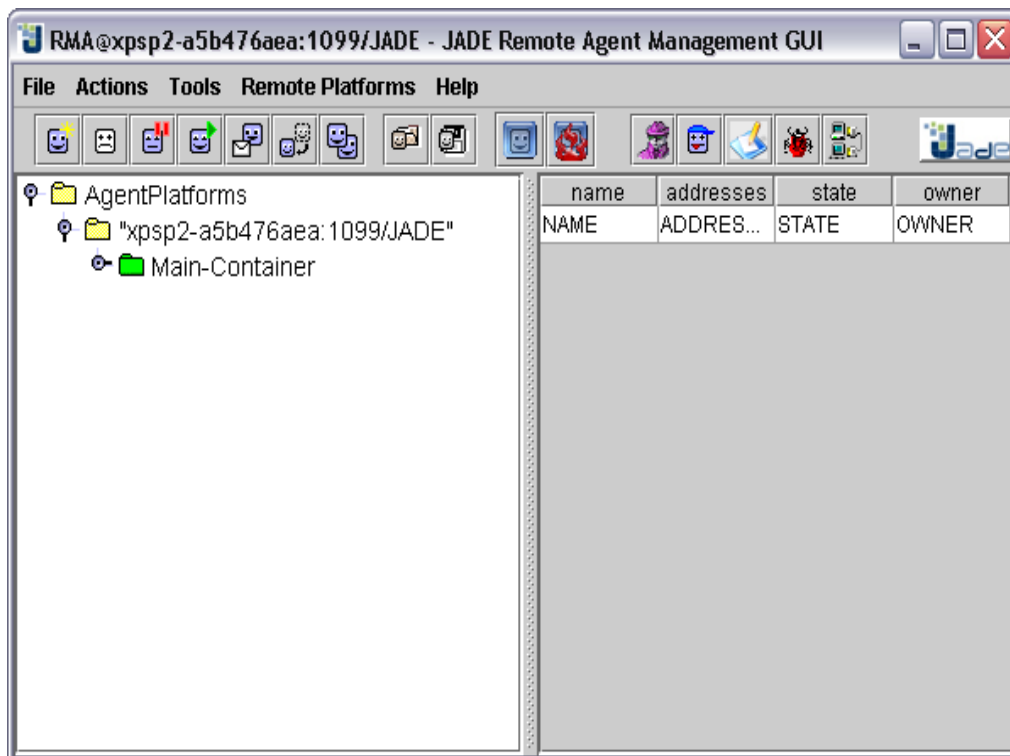


Figure. 3.2 L'interface de la plateforme JADE.

Pourquoi JADE ?

Ces caractéristiques sont naturellement en relation avec nos attentes d'un système multi-agents et des caractéristiques des agents. Ainsi, la plateforme JADE a la propriété suivante: les agents peuvent communiquer entre eux. Cette communication n'est pas un simple échange de messages mais un dialogue. Il ne s'agit pas pour un agent de répondre mécaniquement à un message mais de pouvoir analyser plusieurs possibilités de réponses avant de réagir.

2.3. Création d'un agent

Un agent est créé dans une « place » (Une place est un contexte au sein d'un système d'agent, dans lequel un agent s'exécute). La création peut être initiée soit par un autre agent résidant dans la même place ou par un agent ou un système non agent en dehors de la place. Le créateur doit s'authentifier dans la place et établir les autorisations et les références que va posséder l'agent. La classe de définition nécessaire pour instancier l'agent peut résider dans l'hôte local ou sur une machine distante ou fournie par le créateur.

2.3. Destruction d'un agent

Dans la plupart du temps, l'agent est tué lorsqu'il quitte sa place. Ce processus peut être initié par l'agent lui-même, ou par un autre agent ou système non agent résidant ou non dans la même place.

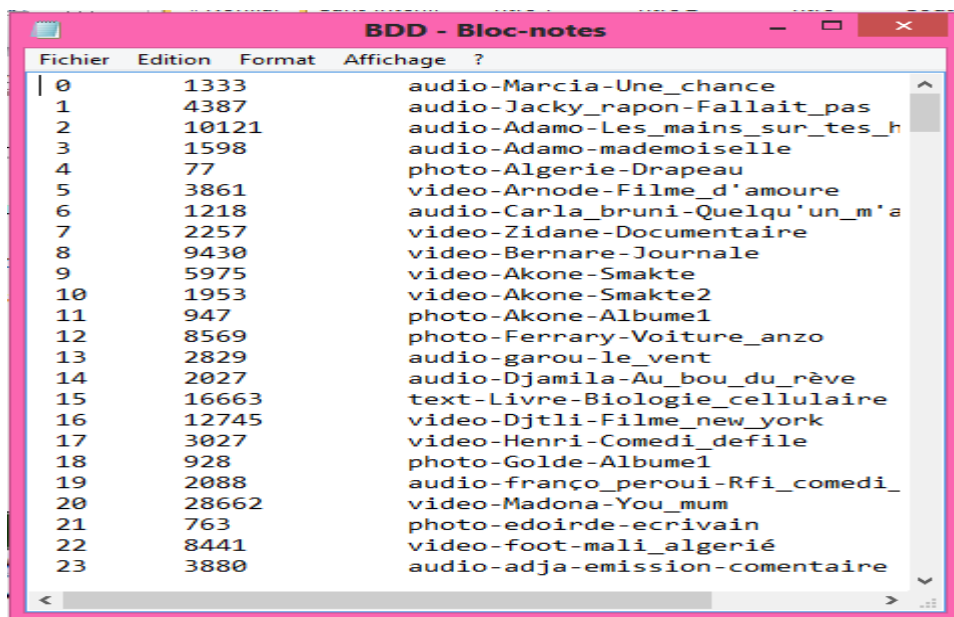
L'agent peut aussi être renvoyé de sa place pour l'une des raisons suivantes:

- La fin de son cycle de vie,
- L'agent n'est pas utilisé ou référencé,
- Violation des règles de sécurité,
- Le système est arrêté.

II. La base de données de notre système de recherche

1. La base de données utilisée

Pour arriver à faire des recherches, nous devons effectuer une simulation de notre système. Pour cela on va utiliser un fichier texte qui va représenter la base de données multimédia de notre application. Ce fichier va contenir un ensemble de ligne, chaque ligne contient un identifiant, la taille, et le nom du document multimédia. Ce fichier texte est unique et n'existe qu'au niveau du serveur.



ID	Size	Filename
0	1333	audio-Marcia-Une_chance
1	4387	audio-Jacky_rapon-Fallait_pas
2	10121	audio-Adamo-Les_mains_sur_tes_h
3	1598	audio-Adamo-mademoiselle
4	77	photo-Algerie-Drapeau
5	3861	video-Arnold-Filme_d'amoure
6	1218	audio-Carla_bruni-Quelqu'un_m'a
7	2257	video-Zidane-Documentaire
8	9430	video-Bernare-Journale
9	5975	video-Akone-Smakte
10	1953	video-Akone-Smakte2
11	947	photo-Akone-Albume1
12	8569	photo-Ferrary-Voiture_anzo
13	2829	audio-garou-le_vent
14	2027	audio-Djamila-Au_bou_du_rève
15	16663	text-Livre-Biologie_cellulaire
16	12745	video-Djtli-Filme_new_york
17	3027	video-Henri-Comedi_defile
18	928	photo-Golde-Albume1
19	2088	audio-franço_peroui-Rfi_comedi_
20	28662	video-Madonna-You_mum
21	763	photo-edoird-ecrivain
22	8441	video-foot-mali_algerié
23	3880	audio-adja-emission-comentaire

Figure 3.3 base de données utilisée

2. Fichier trafic

Dans notre application, le fichier trafic sert à s'intégrer dans un agencement de méthodes de recherche et aussi permet de récupérer l'identifiant de la recherche de chaque client. Il est toujours présent dans chaque machine cliente.

3. Fichier cache

Le cache est une mémoire statique bien plus rapide que la mémoire principale qui permet d'accélérer le fonctionnement d'un système. Dans notre système nous utilisons un

fichier texte comme mémoire cache qui dispose les mêmes mécanismes de fonctionnement que le cache s'intercalant entre le processeur et la mémoire principale.

Vu que la taille du cache est limitée, nous utilisons le mécanisme de remplacement du moins demandé.

III. Modélisation

La modélisation est l'étape la plus importante de toutes études de performances. Elle permet de représenter les différents aspects à étudier en faisant abstraction du système réel. Dans notre application, il s'agit de modéliser un ensemble de clients, un serveur et la procédure de recherche dans le cadre des réseaux sans fil, et ceux en ce terme de taux de succès en requête, le taux de succès en bit, et les nombres de sauts. Notre système d'implémentation est composé d'un ensemble de terminaux mobiles relié tous à un serveur.

Le fonctionnement de notre modèle est :

- ✓ Lorsqu'un client à la recherche de l'information, il va d'abord consulter son propre cache, si le document existe alors il met à jour son cache.
- ✓ Si le document recherché ne se trouve pas dans son cache interne alors il va demander au serveur la liste des clients connectés et choisir une méthode entre le Tabou ou la séquentielle (par la liste d'ordre de connexion des clients) et enfin envoyer la demande au client choisi.
- ✓ Si un client reçoit une requête d'un autre client alors il recherche le document recherché dans son cache. S'il retrouve le document dans son cache alors il renvoi le document trouvé au client demandeur et dans le cas contraire il devient le nouveau chercheur tous en gardant l'adresse du demandeur sur la requête.
- ✓ Pour chaque recherche, le même scénario se réaliser jusqu'à parcourir tous les clients (dans le cas où on ne trouve pas le document). Si le résultat de la recherche est un échec pour tous les clients du réseau mobile alors il envoie la demande au serveur.
- ✓ Lorsque le serveur reçoit la demande, il parcourt la base de données et renvoi le document s'il est dans sa base de données au demandeur.

IV. Mécanismes de recherche

1. Recherche Tabou

La recherche avec tabou a été proposée par Fred Glover en 1986. Depuis cette date, la méthode est devenue très populaire, grâce aux succès qu'elle a remportés pour résoudre de nombreux problèmes. L'intention de l'auteur était de concevoir une méthode de recherche intelligente. La méthode utilise une mémoire (ou plusieurs mémoires) qui sont mises à jour et exploitées au cours de la recherche et elle consiste à choisir un client voisin aléatoirement dans un réseau.

2. Recherche séquentielle

Avec la recherche séquentielle, le client parcourt successivement tous le réseau par l'ordre d'arrive des connectés jusqu'à trouver le document en question.

V. Simulation

1. Interfaces principaux

1.1. Coté client

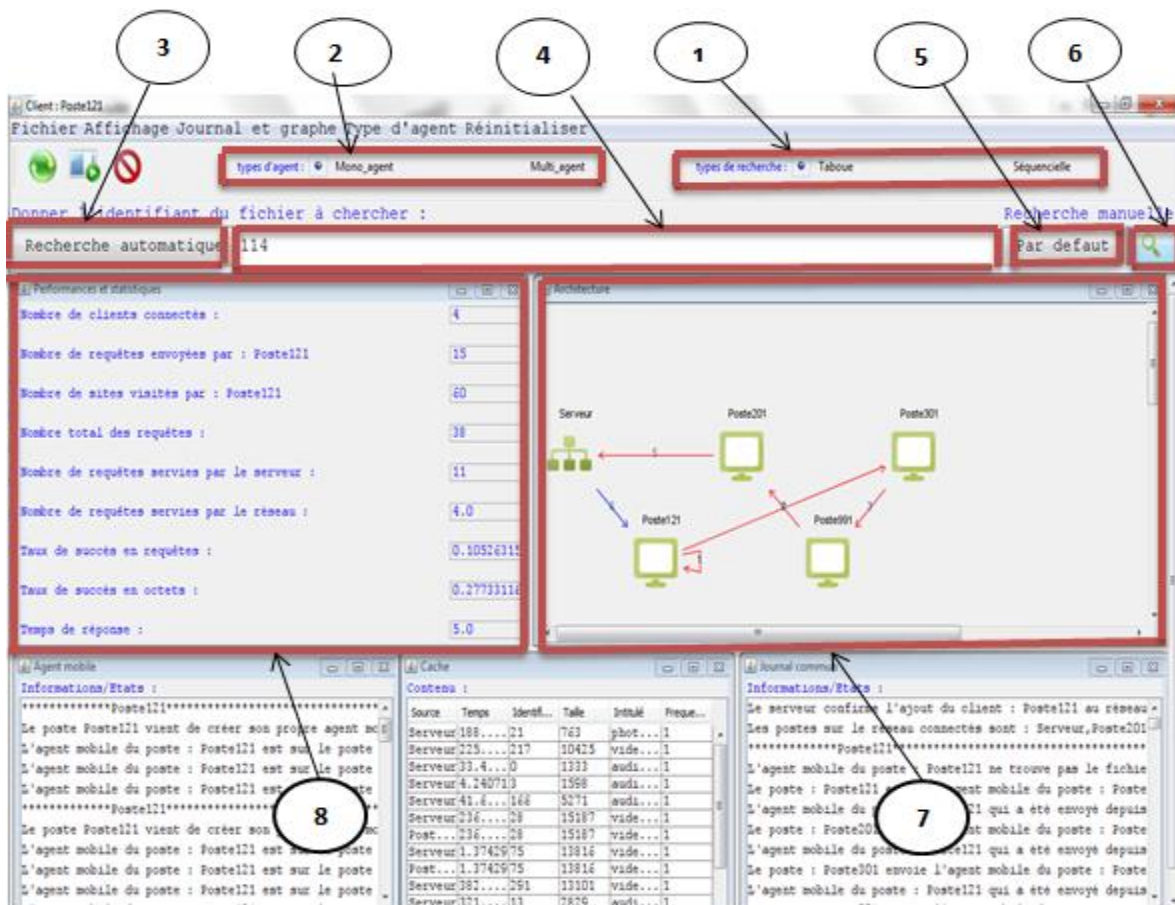


Figure 3.4 Interface de client

- 1- Choix du type de recherche
- 2- Choix du type d'agent
- 3- Lancer une recherche automatique
- 4- Permet la saisie de l'identifiant du document à rechercher
- 5- Introduire un identifiant au hasard
- 6- Permet de faire la recherche de l'identifiant introduit
- 7- Cette fenêtre va afficher l'architecture de la recherche du document
- 8- Cette fenêtre va afficher les performances et les statistiques après l'exécution de la recherche

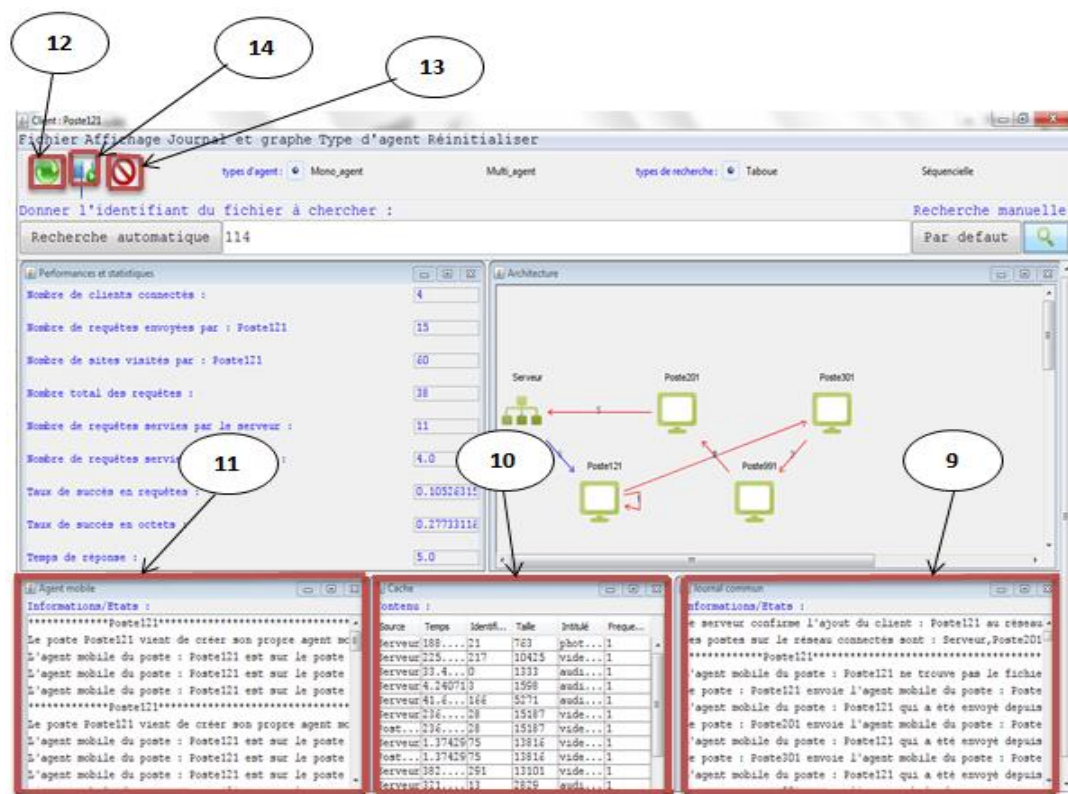


Figure 3.5 Interface de client

- 9- Cette fenêtre va afficher les informations de tous les clients
- 10- Cette fenêtre va afficher le contenu du cache local
- 11- Cette fenêtre va afficher le fonctionnement de l'agent
- 12- Ce bouton sert à ouvrir le fichier trafic
- 13- Ce bouton sert à fermer la fenêtre client
- 14- Ce bouton sert à ouvrir le fichier intitulé

1.1.1. Menu fichier

Ce menu permet d'ouvrir le fichier trafic, d'ouvrir le fichier intitulé, et aussi de fermer la fenêtre client.

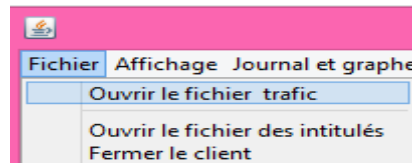


Figure3.6 Menu fichier (coté client)

1.1.2. Menu affichage

Ce menu permet de faire apparaître ou disparaître les fenêtres internes suivantes : journal commun, agent mobile, cache, architecture et performance et statistique.

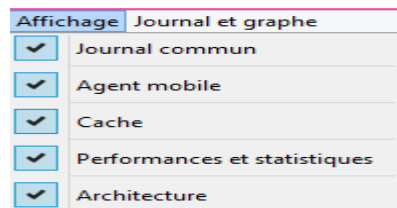


Figure3.7 Menu affichage (coté client)

1.1.3. Menu journal et graphe

Ce menu permet d'afficher l'architecture de chaque client (journal), le graphe de taux de succès en requête, le graphe de taux de succès en octet et le graphe de nombre de saut (temps de réponse).

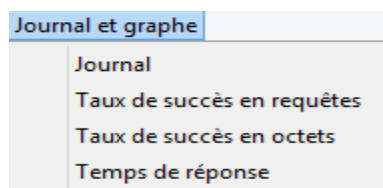


Figure3.8 Menu journal et graphe

1.2. Coté serveur

Voici l'interface obtenue après le lancement de l'application et aussi quelques recherches effectuées

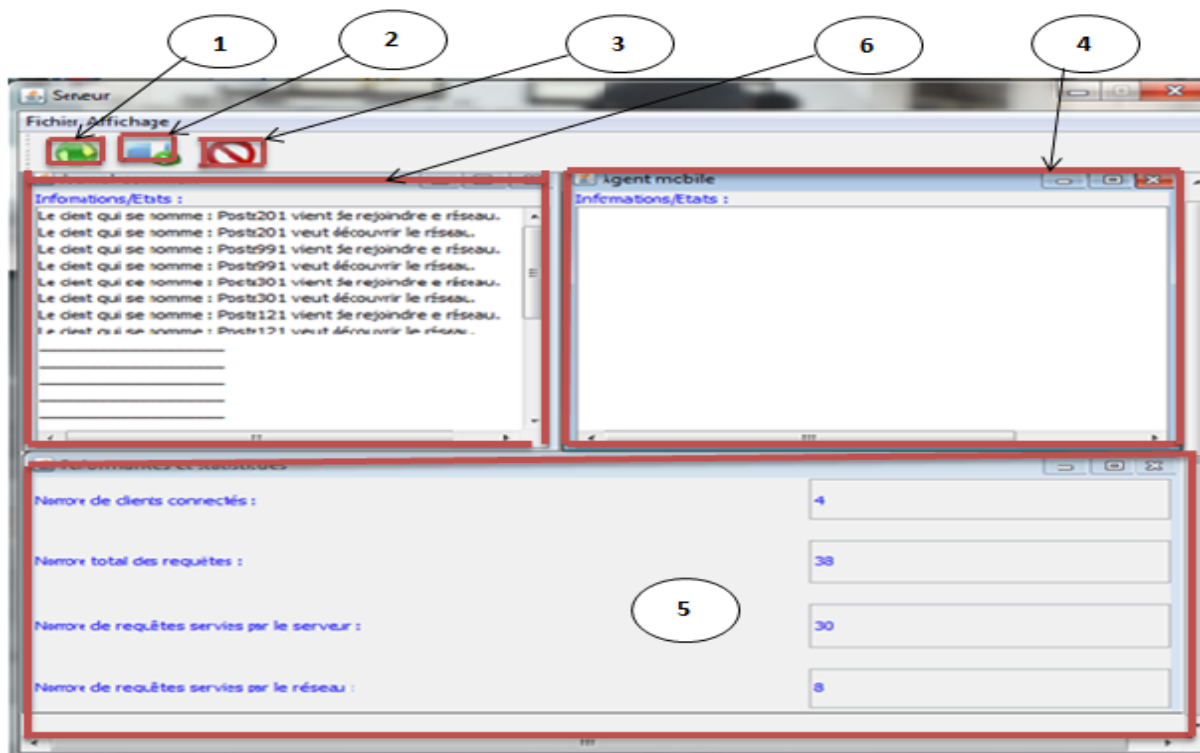


Figure3.9 Interface du serveur

- 1- Ce bouton sert à ouvrir le fichier trafic
- 2-Ce bouton sert à ouvrir le fichier intitulé
- 3- Ce bouton sert à fermer la fenêtre serveur
- 4- Cette fenêtre va afficher le fonctionnement de tous les agents
- 5- Cette fenêtre va afficher les performances et les statistiques après l'exécution de la recherche
- 6- Cette fenêtre nous permet de savoir les clients qui viennent rejoindre le réseau et aussi le client qui a lancé une recherche.

1.2.1. Menu fichier

Ce menu permet d'ouvrir le fichier trafic, d'ouvrir le fichier intitulé, et aussi de fermer la fenêtre serveur.

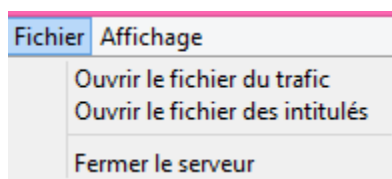


Figure3.10 Menu fichier (coté serveur)

1.2.2. Menu affichage

Ce menu permet de faire apparaître ou disparaître les fenêtres internes suivantes : journal commun, agent mobile et performance et statistique.

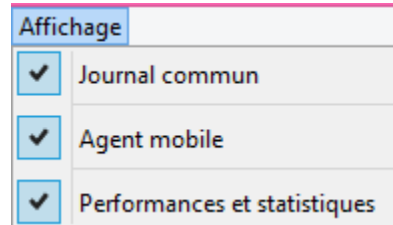


Figure3.11 Menu affichage (coté serveur)

2. Résultats de notre simulation

2.1. Taux de succès en requête (TSR)

- Nombres de requêtes servis par tous les clients (NRSC)
- Nombres de requêtes servis par tous les clients et le serveur (NRSCS)
- $TSR = NRSC / NRSCS$



Figure3.12 taux de succès en requêtes

On remarque que les premières requêtes cherchées ont un taux de succès en requête qui est nul ,cela est dû le fait que tous les caches des clients sont vides et c'est seulement le serveur qui a pu répondre (TSR=0), ensuite la courbe croît vers une certaine valeur cela est dû au fait qu'il y a au moins un client qui a servi la demande d'une requête (TSR≠0), à un certain

moment la courbe décroît jusqu'à une certaine valeur qui est différente de zéro cela veut dire que le serveur a servis à nouveau.

2.2. Taux de succès en octet (TSO)

- ✓ Taille de requêtes servis par tous les clients (TRSC)
- ✓ Taille de requêtes servis par tous les clients et le serveur (NRSCS)
- ✓ $TSO = TRSC / TRSCS$



Figure3.13 *taux de succès en octets*

On remarque que les premières requêtes cherchées ont un taux de succès en octet qui est nul cela est dû au fait que tous les caches des clients sont vides et c'est seulement le serveur qui a pu répondre ($TSO=0$), ensuite la courbe croît vers une certaine valeur, cela est dû au fait qu'il y a au moins un client qui a servi la demande d'une requête ($TSO \neq 0$), à un certain moment la courbe décroît jusqu'à une certaine valeur qui est différente de zéro cela veut dire que le serveur a servis à nouveau.

2.3. Le nombre de saut

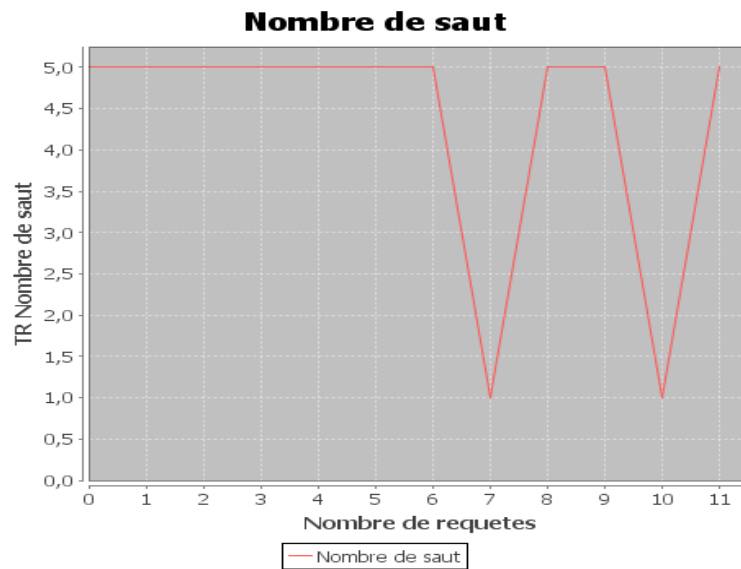


Figure3.14 le nombre de saut

A travers cette courbe, on constate que les premières requêtes cherchées ont un nombre de saut maximal car c'est le serveur qui a servis, à un certain moment la courbe décroît jusqu'à une certaine valeur qui est différente de zéro cela signifie que c'est l'un des clients qui a servis et la courbe croît jusqu'à au nombre de saut maximal cela veut dire que le serveur a servis à nouveau sinon c'est l'un des clients qui a servi.

2.4. Comparaison de deux types de recherches

A partir des graphes précédentes, on peut dire que la recherche Tabou est meilleure car pendant l'exécution de cette recherche, le nombre de sites visités est moins élevé ceci signifie que le temps de réponse est minime par contre dans la recherche séquentielle, le nombre de sites visités est plus élevé avec un temps de réponse plus élevé.

Conclusion

A partir de ce chapitre, l'utilisateur peut avoir une idée à propos du langage java et d'avoir une connaissance au niveau de la plateforme JADE.

D'après les mécanismes de recherches (recherche tabou et séquentielle) que nous avons utilisés dans les réseaux mobiles pour la recherche d'un document, nous pouvons en déduire que la recherche tabou fourni plus de performance car le nombre de sites visités est moins élevé que celui de la recherche séquentielle.

Conclusion générale

L'accroissance des réseaux mobiles est partout dans le monde car elle offre un meilleur moyen aux utilisateurs d'effectuer leurs recherches mais parfois elle pose de problèmes au moment de la recherche tels que la surcharge de serveur, la disponibilité des données ; il est important qu'à chaque fois un client désire avoir une information, de l'avoir mais d'une manière rapide. Il y a beaucoup de solutions qu'on peut réaliser pour résoudre ces problèmes telles que le développement des protocoles de routages dynamiques qui peuvent améliorer la connectivité des nœuds mobiles, la coopération entre les caches.

Dans notre travail, la solution que nous avons choisie pour résoudre ces genres de problème est basée sur la coopération entre les caches pour la gestion et le partage des données multimédias. Nous avons donc implémenté deux types de recherche dans les réseaux mobiles, la recherche Tabou basée sur la sélection aléatoire des machines voisines dans le réseau mobile et la recherche séquentielle basée sur la sélection par ordre de connexion des clients dans le réseau mobile.

La comparaison des résultats des deux types de recherches montre que tous ces deux types diminuent la surcharge du serveur, offre une gestion optimale de la bande passante et permettent un accès rapide aux données grâce à la distribution de la base de données. Ensuite nous avons constaté que la recherche Tabou donne de meilleurs résultats parce que il y a moins de sites visités pour avoir les résultats.

Ce qui conclut que la recherche Tabou est meilleure que la recherche séquentielle.

En conclusion, nous sommes conscient que des efforts restent à faire afin d'arriver à de meilleurs résultats mais nul doute que la créativité qui est propre au multimédia saura ouvrir de nouvelles portes et qu'elle sera un moteur d'expansion et de maturation pour l'informatique du futur.

Bibliographie

- [1] **E. F. Codd** « A Relational Model of Data for Large Shared Data Banks », *CACM* 13, n° 6, June 1970»
- [2] www.cmi.univ-mrs.fr « Cours de bases de données relationnelles»13-juin-11
- [3] **S.Spaccapietra, C.Vingenot**, Bases de données réparties,
http://lbdwww.epfl.ch/f/teaching/courses/slidesBDA/BDR/BDR_se.pdf
- [4] URL: <http://ceria.dauphine.fr/Rim/SupportBDR.pdf>
- [5] **DI GALLO Frédéric**, « WiFi, L'essentiel qu'il faut savoir... », Extraits de source diverses récoltées en 2003, P5.
- [6] **Mark L. Gillenson. 2004.** «Fundamentals of Database Management Systems. » Wiley E-Books. www.wiley.com/
- [7] <http://www.techno-science.net/?onglet=glossaire&definition=316>
- [8] Microsoft. Checking Control Access Right-Based Access. [http://msdn.microsoft.com/en-us/library/cc223518\(v=prot.10\).aspx](http://msdn.microsoft.com/en-us/library/cc223518(v=prot.10).aspx).
- [9] **L. Fan, P. Cao, J. Almeida, A. Z. Broder**, *Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol*, IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 8, NO. 3, JUNE 2000
- [10] **Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall and G.J. Minden**, "A Survey of active network research," IEEE communications Magazine, pp.80--86, January 1997.
- [11] **P. Cao and S. Irani**, *Cost-Aware WWW Proxy Caching Algorithms*, Proceedings of the 1997 USENIX Symposium on Internet Technology and Systems, pp. 193-206, Dec 1997.
- [12] **A.Belloun and L.Hertzberger**.Dealing with onetimer document in web caching, 1998.
- [13] **G. Barish and K. Obraczka**, "World Wide Web Caching: Trends and Techniques," IEEE Communications, May 2000.
- [14] **J. S. Gwertzman and M. Seltzer**. *The case for geographical push-caching*. In Proceedings of the Workshop on Hot Topics in Operating Systems, pages 51-57, May 1995.
- [15] **A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrel**, *A hierarchical Internet object cache*, Usenix'96, January 1996.
- [16] **Duane Wessels, k. claffy** *ICP and the Squid Web Cache* August 13, 1997
- [17] **A. Rousskov and D. Wessels**, "Cache Digest" Proceedings of the WCW'98, Manchester, England, 1998.

- [18] **Víctor J. Sosa, Leandro Navarro**, *Influence of the Document Validation/Replication Methods on Cooperative Web Proxy Caching Architectures* (2002).
- [19] **L. Zhang, S. Floyd, and V. Jacobson**. *Adaptive Web Caching*. In Proceedings of the NLANR Web Cache Workshop, June 1997. 14.
- [20] **P. Rodriguez, C. Spanner, E.W. Biersack**, "Web Caching Architectures: Hierarchical and Distributed Caching". 4th International Caching Workshop, 1999.
- [21] **L. A. Belady, R. A. Nelson, and G. S. Shedler**. An anomaly in space-time characteristics of certain programs running in a paging machine. *Communications of the ACM*, 12(6) :349–353, 1969.
- [22] **Peter Scheuermann, Junho Shim, and Radek Vingralek**. Watchman: A data warehouse intelligent cache manager. In The International Conference on Very Large Data Bases, pages 51–62, 1996.
- [23] **Yigal Arens and Craig A. Knoblock**. Intelligent caching: selecting, representing, and reusing data in an information server. In The international conference on Information and knowledge management, pages 433–438, 1994.
- [24] **Steven Glassman**. A caching relay for the World Wide Web. In The International World Wide Web Conference, pages 165–173, 1994.
- [25] **J. Ferbert, O. Gutknecht & F. MICHEL**, «Agent-Oriented Software Engineering», IV the International Workshop (AOSE), 2935, pp 214-230. Springer Verlag, Australia, mars 2004.
- [26] **J. TISSEAU, G. PRIGENT**, « Simulation de la disponibilité des systèmes d'information -Etude & Réalisation d'une plateforme de simulation multi-agents », Centre Européen de Réalité Virtuelle, France, 16 juin 1999.
- [27] **Olivier Fourdrinoy** Maîtrise d'informatique, Faculté Jean Perrin, Université d'Artois, Lens.
- [28] **Rimassa G., Bellifemine F., Poggi A.**, JADE - A FIPA Compliant Agent Framework, *PMAA '99*, p. 97-108, Londres, Avril 1999.