6^{emes} **journées d'Extraction et de Gestion des Connaissances (EGC'2006)**

17-20 janvier 2006, Lille

Deuxième atelier

Extraction et Gestion Parallèles Distribuées de Connaissances (EGPDC'2006)

Organisé par : N. Melab & E-G. Talbi, LIFL, Université de Lille1

Deuxième atelier sur

Extraction et Gestion Parallèles Distribuées de Connaissances (EGPDC'2006)

Associé à EGC'2006

Présentation

L'évolution technologique en matière de moyens de stockage et de puissance de calcul des machines a conduit à l'émergence de grandes masses de données. Très souvent, ces volumes de données sont répartis et disséminent des connaissances insoupçonnées. Encore, faut-il disposer de méthodes et d'outils parallèles et/ou distribués capables d'extraire ces connaissances pertinentes, de les stocker, et de les diffuser. Aussi, de nombreuses activités de recherche (ACI Masse de données, ACI IMPBio, ACI Grid, etc.) se développent autour de l'extraction et de la gestion parallèles distribuées de connaissances.

L'objectif de cet atelier est de rassembler les chercheurs des disciplines connexes (systèmes parallèles distribués, systèmes d'information et bases de données distribués, extraction de connaissances, etc.). L'atelier sera ainsi l'occasion d'échanger les idées et faire l'état de l'art des problématiques liées à l'extraction et à la gestion parallèles distribuées de connaissances.

Thèmes majeurs (liste non limitative)

- Modèles de représentation des données distribuées et des connaissances.
- Méthodes de prétraitement de données volumineuses distribuées.
- Méthodes d'accès aux données distribués.
- Algorithmes parallèles distribués d'extraction de connaissances à partir de sources hétérogènes.
- Entrepôts de données distribués : construction, exploitation, maintenance, etc.
- Méthodes de collecte, d'agrégation et de synthèse de connaissances partielles.
- Gestion de la sécurité et de la confidentialité des données distribuées et des connaissances.
- Intergiciels pour l'extraction et la gestion parallèles distribuées des connaissances.
- Grilles et systèmes pair-à-pair pour l'extraction et la gestion des connaissances.
- Applications du monde réel : médecine, télécommunications, génomique, administration, banque, etc.

Organisateurs

Nouredine Melab (melab@lifl.fr) & El Ghazali Talbi (talbi@lifl.fr)
LIFL, CNRS UMR 8022, Université de Lille1
INRIA Futurs

Comité de lecture

L. Brunie (LIRIS, INSA de Lyon)
A. Hameurlain (IRIT, Université Paul Sabatier, Toulouse)
N. Melab (LIFL, Université de Lille1)
J-M. Pierson (LIRIS, INSA de Lyon)
E-G. Talbi (LIFL, Université de Lille1)
A. Tchounikine (LIRIS, INSA de Lyon)
B. Toursel (LIFL, Université de Lille 1)

TABLE DES MATIÈRES

Apport des agents mobiles à l'accès efficace aux données distribuées à grande échelle.
Mohammad Hussein, Abdelkader Hameurlain, Franck Morvan
IRIT, Université Paul Sabatier, Toulouse 1

Optimisation combinatoire sur grilles de calcul pour la fouille de données en spectroscopie PIR.
Nouredine Melab, Sébastien Cahon, El-Ghazali Talbi
CNRS/LIFL, Université de Lille1 9

NDS : Network Distance Service - un outil de calcul de distances réseau noeud à noeud.
Julien Gossa, Jean-Marc Pierson
LIRIS, INSA Lyon 18

Gestion de la Sécurité pour l'Extraction Parallèle Distribuée des Connaissances.
Serge Chaumette (LaBRI, Université de Bordeaux 1), Achraf Karray (ENIS, Université de Sfax, Tunisie), Damien Sauveron (XLIM, Université de Limoges) 24

Apport des agents mobiles à l'accès efficace aux données distribuées à grande échelle [1]

Mohammad Hussein, Abdelkader Hameurlain, Franck Morvan

Institut de Recherche en Informatique de Toulouse IRIT, Université Paul Sabatier
118, Route de Narbonne, 31062 Toulouse Cedex, France
Phone : 33 (0) 5 61 55 82 48, Fax: 33 (0) 5 61 55 62 58
E-mail : {hussein, hameur, morvan}@irit.fr

Résumé. L'objectif de cet article est de mettre en évidence l'apport des agents mobiles à l'accès efficace aux données volumineuses, hétérogènes, et réparties à grande échelle. Après avoir donné des exemples d'application, notamment ceux issus des bases de données biomédicales, nous décrivons un modèle d'exécution à base d'agents mobiles pour l'optimisation de requêtes réparties à grande échelle. Nous montrons, ensuite, comment les agents mobiles peuvent s'adapter dynamiquement aux erreurs d'estimation, à l'instabilité de l'environnement d'exécution, et à l'indisponibilité des ressources. Des expériences menées en environnements local et à grande échelle, nous ont permis : (i) de valider la politique de migration proactive proposée, et (ii) d'identifier des intervalles d'efficacité des agents mobiles. Enfin, dans le cadre du projet Grille Géno-Médicale GGM (ACI Masses de Données 2004), nous étudions, actuellement, comment ce modèle d'exécution à base d'agents mobiles peut être déployé sur une grille de calcul afin d'offrir un service de requêtes optimisées.

1 Introduction

Avec le succès et l'évolution rapide de la technologie des réseaux, le nombre de sources de données accessibles aux travers ces réseaux ne cesse de croître. Cette croissance a donné naissance à de nouvelles applications référençant différentes sources de données. En effet, les données relatives à un sujet sont souvent éparpillées sur plusieurs sources autonomes et hétérogènes. Par exemple, considérons qu'un utilisateur de la caisse primaire d'assurance maladie recherche le numéro de téléphone du médecin traitant d'un patient identifié par son numéro de sécurité sociale. Ceci nécessite l'accès au système d'information de la sécurité sociale pour récupérer le nom du médecin traitant et son département. Puis, à partir de ces informations, l'utilisateur interroge le service des pages jaunes accessible via Internet pour récupérer le numéro de téléphone du médecin. Un autre exemple d'application est la corrélation entre les gènes humains et les maladies génétiques. Par exemple, considérons la requête qui consiste à rechercher les fonctions moléculaires des gènes impliqués dans la maladie d'Alzheimer. Pour produire le résultat de cette requête, il faut tout d'abord identifier les gènes à la base de la maladie d'Alzheimer en interrogeant les sources relatives aux

[1] Une partie de ce travail a été soutenu par l'ACI Masses de Données 2004 ; Projet Grille Géno-Médicale GGM, No. 04 2 32.

maladies génétiques comme Gene Reviews (http://www.genetests.org/) et Genes and Disease (http:// www.ncbi.nlm.nih.gov/disease/). Ensuite, les sources de données décrivant les fonctionnalités moléculaires des gènes (e.g. Gene Ontology, http:// www.geneontology.org/) sont interrogées afin de déterminer les fonctionnalités des gènes identifiés.

Pour accéder uniformément aux données provenant de plusieurs sources, des systèmes à base de médiateurs et d'adaptateurs Garcia-Molina et al. (1997), Haas et al. (1999), Manolescu (2001), Wiederhold (1992) (nommés aussi systèmes de médiation de données) ont été conçus et développés. Dans ce contexte, traiter efficacement les requêtes présente de grandes difficultés. En effet, un plan d'exécution engendré par un optimiseur classique de requêtes réparties, peut présenter de mauvaises performances pour quatre raisons principales : (i) la centralisation de l'optimisation, (ii) l'imprécision des estimations, (iii) l'indisponibilité des ressources (i.e. données, CPU, bande passante du réseau, et mémoire). Pour réagir à tous ces événements, des méthodes d'optimisation dynamique de requêtes Amsaleg et al. (1998), Bouganim et al. (2000), Ives et al. (2004), Zhou et al. (2005) ont été développées. Ces méthodes corrigent, en cours d'exécution, les sous-optimalités des plans d'exécution. La majorité des méthodes d'optimisation dynamique proposées à ce jour sont centralisées. Cette centralisation engendre un goulet d'étranglement et produit un échange de messages relativement important sur un réseau à faible débit et à forte latence. Ainsi, il devient important de rendre autonome et auto-adaptable l'exécution des requêtes. Dans cette perspective, une approche à explorer consiste à s'appuyer sur le modèle de programmation à base d'agents mobiles Fuggetta (1998). Un agent mobile est une entité logicielle autonome et adaptable, capable de se déplacer dynamiquement (code, données et état d'exécution) pour accéder à des données ou à des ressources distantes. Il est important de rappeler qu'actuellement, les plates-formes d'agents mobiles ne fournissent que des mécanismes de déplacement (i.e. migration), mais n'offrent pas de politique de décision de déplacement.

Dans cet article, nous présentons un modèle d'exécution, incluant une politique de décision proactive, à base d'agents mobiles pour l'optimisation dynamique de requêtes réparties à grande échelle. Dans ce modèle, chaque opérateur relationnel est exécuté au moyen d'un agent mobile. Un agent mobile réagit, en cours d'exécution, aux erreurs d'estimation, à l'indisponibilité des ressources et à l'instabilité de l'environnement d'exécution pour éventuellement migrer d'un site à un autre site pour continuer son exécution. Ainsi, pour qu'un agent puisse migrer d'une manière autonome et sans être dépendant de l'optimiseur, il s'appuie sur un modèle de coûts embarqué dans un agent.

Le reste de l'article est organisé comme suit : la section 2 a pour objective de proposer une synthèse des principales méthodes d'optimisation dynamique de requêtes. La section 3 décrit notre modèle d'exécution à base d'agents mobiles pour l'optimisation dynamique de requêtes. La section 4 est consacrée à l'évaluation des performances de notre modèle d'exécution à base d'agents mobiles. Enfin, la section 5 établit une conclusion, récapitule les problèmes ouverts et décrit les contours des travaux futurs.

2 Etat de l'art

Cette section décrit une courte synthèse d'un ensemble des méthodes d'optimisation dynamiques Hussein (2005b). Une méthode d'optimisation est dite dynamique si elle possède les trois caractéristiques suivantes Hellerstein et al. (2000) : (i) elle reçoit des informations à partir de son environnement, (ii) elle utilise ces informations pour déterminer

son comportement, et (iii) elle exécute les étapes (i et ii) d'une manière itérative, ce qui permet à une méthode d'optimisation dynamique d'adapter les plans d'exécution aux changements de l'environnement d'exécution.

Les méthodes d'optimisation dynamique peuvent être classifiées en deux grandes catégories : les méthodes d'optimisation dynamique centralisées Amsaleg et al. (1998), Bouganim et al. (2000), Evrendilek et al. (1997), Ives et al. (2004), Zhou et al. (2005) et les méthodes d'optimisation dynamique décentralisées Collet et Vu (2004), Ives et al. (1999), Jones et Brown (1997), Urhan et Franklin (2000). Dans les méthodes d'optimisation dynamique centralisées, le contrôle et les modifications du plan d'exécution sont effectués par un optimiseur centralisé. Cet optimiseur collecte, en cours d'exécution, des informations concernant les statistiques sur les données, les ressources disponibles et les coûts de communication. Ensuite, à partir des informations collectées, l'optimiseur décide si un plan doit être modifié ou non. Cependant, la centralisation ne permet pas à ces méthodes de passer à l'échelle à cause : (i) du nombre de messages relativement important sur un réseau à faible débit et à forte latence, et (ii) du goulet d'étranglement que forme l'optimiseur puisque tous les messages convergent vers un seul point. Pour cela, dans un environnement réparti à grande échelle le contrôle et les modifications des plans d'exécution doivent être décentralisés.

La décentralisation du contrôle et les modifications des plans d'exécution dans les méthodes d'optimisation dynamique évitent le goulet d'étranglement formé par la centralisation. La méthode proposée par Collet et Vu (2004) s'appuie sur les négociateurs pour rendre autonomes, dynamiques et décentralisées les exécutions des sous-requêtes locales appartenant à une même requête répartie. Dans les méthodes proposées par Urhan et Franklin (2000), Ives et al. (1999), les opérateurs de jointure sont autonomes, dynamiques et décentralisés afin de réagir aux changements de l'environnement d'exécution. Ces méthodes Collet et Vu (2004), Ives et al. (1999), Urhan et Franklin (2000) améliorent le coût de traitement local en adaptant l'utilisation des ressources CPU, E/S et mémoire aux changements de l'environnement d'exécution (e.g. erreurs d'estimation, délais imprévus d'arrivées de données). Cependant, les méthodes proposées se focalisent principalement sur les ressources (CPU, E/S et mémoire) et ne considèrent pas la ressource réseau. Les transferts de données déterminés lors de la compilation restent identiques quelles que soient les erreurs d'estimation constatées lors de l'exécution.

La méthode proposée par Jones et Brown (1997) s'appuie sur les agents mobiles pour décentraliser le contrôle et les modifications des plans d'exécution. Elle améliore le coût de traitement local et le coût de communication en minimisant le volume de données transférées sur le réseau. Cette méthode se base sur la diffusion de messages entre les différents agents exécutant la même requête pour détecter les sous-optimalités et pour les corriger. Cette diffusion des messages de contrôle engendre une congestion du réseau lorsque le nombre d'agents devient important. Dans un environnement à grande échelle, les méthodes d'optimisation dynamique en s'adaptant aux changements de l'environnement d'exécution doivent minimiser le volume de données transférées tout en minimisant les *messages de contrôle*. Ce dernier permet d'éviter que le coût induit par ces messages soit plus grand que le coût économisé par la réduction du volume de données transférées. Dans ce cadre, l'utilisation des agents mobiles dans l'optimisation de requêtes réparties à grande échelle nous semble une voie prometteuse. En effet, les agents mobiles permettent : (i) de décentraliser le contrôle et les modifications des plans d'exécution, (ii) de corriger les sous-

optimalités en tenant compte à la fois du coût de traitement local et du coût de communication et (iii) de remplacer les interactions distantes par des interactions locales.

3 Modèle d'exécution pour l'optimisation de requêtes à base d'agents mobiles

Dans cette section, nous décrivons un modèle d'exécution pour l'optimisation dynamique de requêtes réparties en s'appuyant sur le paradigme d'agents mobiles. Nous commençons par l'introduction des briques de base qui sont les opérateurs de jointure mobile. Ensuite, nous présentons un modèle d'exécution à base d'agents mobiles pour les requêtes. Enfin, nous définissons un modèle de coûts embarqué dans un agent mobile.

3.1 Jointures mobiles

Pour réagir aux changements imprévus en environnement réparti à grande échelle, il est opportun de rendre, chaque opérateur relationnel (e.g. jointure) autonome et capable de choisir l'endroit où il s'exécute en s'appuyant sur les agents mobiles. Avec la mobilité logicielle des agents, les algorithmes de jointure directe et à base de semi-jointure ont été étendus afin de permettre à ces jointures directes et à base de semi-jointure de changer proactivement leurs sites d'exécution Arcangeli et al. (2004). La décision et le contrôle du changement du site d'exécution sont effectués d'une manière décentralisée et autonome. Ce n'est plus l'optimiseur qui choisit le site d'exécution de la jointure, mais c'est la jointure elle-même qui choisit son site d'exécution. En effet, l'agent exécutant une jointure s'adapte à la variation des caractéristiques de l'environnement d'exécution. Ainsi, il réagit, en cours d'exécution, aux erreurs d'estimation (e.g. taille des relations intermédiaires), à l'indisponibilité des ressources et à l'instabilité de l'environnement d'exécution pour éventuellement migrer d'un site à un autre. Pour qu'un agent puisse migrer d'une manière autonome et sans être dépendant de l'optimiseur, il a besoin d'une politique de migration proactive qui lui permet de choisir son site d'exécution le plus adéquat. Cette politique de migration doit tenir compte non seulement des erreurs d'estimation mais aussi de l'indisponibilité des ressources sollicitées (i.e. données, CPU, mémoire, et bande passante du réseau).

3.2 Requêtes mobiles

Après avoir introduit la notion de la jointure mobile, nous présentons dans cette sous-section un modèle d'exécution dynamique et autonome pour les requêtes. Dans ce modèle, chaque opérateur du plan d'exécution est exécuté au moyen d'un agent mobile. Dans le cas où un agent participe à l'exécution d'une requête (i.e. il fait partie d'un ensemble d'agents qui exécutent une même requête), sa décision de migrer ne peut être prise sans tenir compte des décisions prises par les autres agents communiquant avec lui. En effet, la décision isolée d'un agent lui permet d'améliorer le temps de réponse de son opérateur mais ne permet pas de garantir la minimisation du temps de réponse de la requête entière. Par exemple, considérons deux jointures J1 : T= Jointure (R1, R2) et J2 : res=Jointure (T, R3) s'exécutant respectivement au moyen de deux agents mobiles AJ1 et AJ2. Les agents AJ1 et AJ2 sont placés respectivement sur les sites S1 et S2. Supposons, maintenant que AJ1 décide de

migrer sur S2 pour se rapprocher de AJ2 et qu'en même temps AJ2 décide de migrer sur S1 pour les mêmes raisons. Nous obtenons, ainsi, par manque de coopération entre AJ1 et AJ2, une décision contraire aux objectifs de rapprochement de AJ1 et AJ2.

Pour éviter les mauvaises décisions prises par un agent, nous avons proposé trois méthodes de coopération Morvan et al. (2003). Ces méthodes de coopération étant réalisées en environnement réparti à grande échelle, elles doivent minimiser au maximum le nombre de communications de données et de contrôles entre les agents. La première, nommée coopération hiérarchique (CH), permet aux agents communiquant entre eux de connaître leurs sites d'exécution respectifs. La deuxième qui est appelée coopération hiérarchique et propagation des estimations révisées (CHPE), ajoute à la première une propagation partielle des estimations révisées lors de l'exécution concernant les relations temporaires. Quant à la troisième, elle propage les estimations révisées avant et pendant l'exécution de la jointure mobile. Cette méthode, permettant à une jointure mobile de migrer sans ses données, est nommée double coopération hiérarchique avec propagation des estimations (DCHPE).

3.3 Modèle de coûts embarqué dans un agent

Nous rappelons qu'un agent mobile exécutant un opérateur mobile peut migrer d'un site à un autre afin d'améliorer le temps de réponse de son opérateur. Le site de migration est choisit par un agent en fonction de plusieurs métriques : les coûts de production locale des opérandes, le coût d'exécution locale de l'opérateur d'un agent, le coût de migration d'un agent, les coûts de transfert des opérandes du site de localisation des opérandes vers le site d'exécution de l'opérateur, et le coût de transfert du résultat. Ces métriques sont calculées par le médiateur du site où l'agent est localisé. Le problème ici est que le catalogue du modèle de coûts du médiateur où l'agent est localisé ne contient pas, forcément, toutes les informations nécessaires pour calculer les métriques d'un agent. Ainsi, pour assurer l'autonomie d'un agent mobile tout en évitant des interactions distantes, nous avons défini un modèle de coûts embarqué dans un agent mobile Hussein et al. (2005a). Ce modèle de coûts est constitué d'un espace de migration (i.e. ensemble de sites où un agent peut migrer), des profils des opérandes et des valeurs des paramètres de coûts. Le modèle de coûts embarqué dans un agent et le modèle de coûts du médiateur où un agent est localisé, permettent à un agent de prendre sa décision concernant le choix de son site d'exécution.

4 Evaluation des performances

Pour valider nos propositions, nous avons développé une plate-forme d'expérimentation s'appuyant sur : une plate-forme d'agents mobiles Arcangeli et al. (2002), le langage JAVA et l'intergiciel RMI. En utilisant cette plate-forme d'expérimentation, nous avons mesuré, par exécution, les temps de réponse de la jointure mobile par hachage en environnement local et en environnement à grande échelle. A partir de ces expériences, nous pouvons faire les observations suivantes Hussein et al. (2005a) :

(i) dans nos environnements d'expérimentation, les temps de réponse de la jointure mobile par hachage sont meilleurs que ceux de la jointure par hachage classique lorsqu'une erreur de sur-estimation est constatée (20% d'erreur sur la taille du premier opérande de la jointure évaluée ou 30% d'erreur sur le facteur de sélectivité),

(ii) la charge du site d'exécution influe significativement sur le temps de réponse d'une jointure mobile en environnement local alors que son influence est relativement faible en environnement à grande échelle.

Quant à l'évaluation des performances de requêtes mobiles, nous nous sommes appuyés sur un modèle de simulation calibré, puisque nous ne possédons pas, actuellement, assez de sites d'exécution reliés par un réseau à grande échelle. En utilisant ce modèle de simulation, nous avons comparé les performances des trois méthodes de coopération proposées en considérant les 3 structures arborescentes (i.e., arbre linéaire gauche, arbre linéaire droit, et arbre ramifié) associées à la requête multi-jointure $R_1 \infty R_2 \infty R_3 \infty R_4$. L'évaluation des performances de ces méthodes nous a permis d'observer la supériorité de la méthode DCHPE par rapport aux méthodes CH et CHPE. De plus, nous remarquons que la méthode DCHPE possède un comportement homogène quelle que soit la structure arborescente du plan d'exécution utilisé dans nos évaluations. Cependant, nous constatons un léger surcoût de la méthode DCHPE lorsque que la sous-estimation ou la sur-estimation de R1, faite par l'optimiseur, est faible (plus ou moins 30% d'erreur sur la taille de R1).

5 Conclusion et perspectives

Dans cet article, nous avons décrit un modèle d'exécution à base d'agents mobiles pour l'optimisation de requêtes réparties à grande échelle. Dans ce modèle, chaque opérateur est exécuté au moyen d'un agent mobile. La mobilité des agents permet de changer le site d'exécution de l'opérateur. Nous avons proposé également trois méthodes de coopération permettant à un agent évaluant un opérateur d'une requête de changer son site d'exécution en tenant compte des décisions des autres agents. Le modèle de coûts embarqué dans un agent garantit son autonomie pendant son exécution. De plus, il permet à un agent de réagir aux changements de l'environnement d'exécution sans retourner ou invoquer l'optimiseur où la requête a été soumise.

Les prochains objectifs de nos travaux se focaliseront, sur la définition, des méthodes permettant de déterminer les espaces de migration des agents participant à l'exécution d'une même requête. De plus, à partir de l'espace de migration d'un agent, il est nécessaire de définir des mécanismes déterminant le site d'exécution initial d'un agent. Enfin, nous étendons, actuellement, notre plate-forme d'expérimentation pour compléter l'évaluation des performances des requêtes mobiles. En effet, dans le cadre du projet GGM (Grille Géno-Médicale, http://liris.cnrs.fr/PROJETS/liris/projets/ggm) de l'ACI Masse de données 2004, nous sommes entrain de construire une mini-grille permettant de déployer un service de requêtes optimisées. Dans ce contexte, nous mènerons de nouvelles expériences plus exhaustives afin d'étudier les comportements des agents exécutant des requêtes.

Références

Amsaleg, L., M. Franklin et A. Tomasic (1998). *Dynamic query operator scheduling for wide-area remote access,* Distributed and Parallel Databases, vol. 6, n°3, Kluwer Academic Publishers, 217-246.

Arcangeli, J. P., A. Marcoux, C. Maurel et F. Migeon (2002). *Programming mobile and adaptable agents with JavAct (Version 3)*. Rapport de recherche, 2002-37-R, Université Paul Sabatier, Toulouse, IRIT.

Arcangeli, J. P., A. Hameurlain, F. Migeon et F. Morvan (2004). *Mobile Agent Based Self-Adaptive Join for Wide-Area Distributed Query Processing*, Journal of Database Management, Idea Group, vol. 15, n°4, 25-44.

Bouganim, L., F. Fabret, C. Mohan et P. Valduriez (2000). *Dynamic query scheduling in data integration systems*, Proc. of the 16th International Conference on Data Engineering, IEEE Computer Society, San Diego, California, USA, 425-434.

Collet, C. et T. T. Vu (2004). *QBF: A Query Broker Framework for Adaptable Query Evaluatio*n, Proc. of 6th International Conference on Flexible Query Answering Systems, Springer Verlag Publishers, Lyon, France, 362-375.

Evrendilek, C., A. Dogac, S. Nural et F. Ozcan (1997) *Multidatabase Query Optimization*, Journal of Distributed and Parallel Databases, Kluwer Academic Publishers, vol 5, n°1, 77-113.

Fuggetta, A., G. P. Picco et G. Vigna (1998). *Understanding Code Mobility*, IEEE Transactions on Software Engineering, IEEE Computer Society, vol. 24, n°5, 342-361.

Garcia-Molina, H. et al. (1997). *The TSIMMIS Approach to Mediation: Data Models and Languages*, Journal of Intelligent Information Systems, Kluwer Academic Publishers, vol. 8, n°2, 117-132.

Haas, L. M. et al. (1999). Transforming Heterogeneous Data with Database Middleware: Beyond Integration, IEEE Data Engineering Bulletin, IEEE Computer Society, vol. 22, n°1, 31-36.

Hellerstein, J. M. et al. (2000). *Adaptive query processing: Technology in evolution*, IEEE Data Engineering Bulletin, IEEE Computer Society, vol. 23, n°2, 7-18.

Hussein M., F. Morvan et A. Hameurlain (2005a). *Embedded Cost Model in Mobile Agents for Large Scale Query Optimization*, Proc. of the 4th International Symposium on Parallel and Distributed Computing, IEEE Computer Society, Lille, France, 199-206.

Hussein, M. (2005b). *Un modèle d'exécution à base d'agents mobiles pour l'optimisation dynamique de requêtes réparties à grande échelle*, Thèse de Doctorat, Spécialité Informatique, Université Paul Sabatier, France.

Ives, Z. G. et al. (1999). *An adaptive query execution system for data integration*, Proc. of the ACM SIGMOD International Conference on Management of Data, ACM Press, Philadelphia, Pennsylvania, USA, 299-310.

Ives, Z. G., A. Y. Halevy et D. S. Weld (2004). *Adapting to Source Properties in Processing Data Integration Queries*, Proc. of the ACM SIGMOD International Conference on Management of Data, ACM Press, Paris, France, 395-406.

Jones, R. et J. Brown (1997). *Distributed query processing via mobile agents*, Retrouvé 14 novembre 2002, accessible via : http://www.cs.umd.edu/~rjones/paper.html.

Manolescu, I. (2001). *Techniques d'optimisation pour l'interrogation des sources de données hétérogènes et distribuées*, Thèse de Doctorat, Spécialité Informatique, Université de Versailles Saint-Quentin-en-Yvlenies, France.

Morvan, F., M. Hussein et A. Hameurlain (2003). *Mobile Agent Cooperation Methods for Large Scale Distributed Dynamic Query Optimization*, Proc. of the 14th International Workshop on Database and Expert Systems Applications, IEEE Computer Society, Prague, Czech Republic, 542-547.

Urhan, T. et M. Franklin (2000). *XJoin : A reactively-scheduled pipelined join operator*, IEEE Data Engineering Bulletin, IEEE Computer Society, vol. 23, n°2, 27-33.

Zhou, Y., B. C. Ooi, K. L. Tan et W. H. Tok (2005). *An adaptable distributed query processing architecture*, Data & Knowledge Engineering, vol. 53, n°3, 283-309.

Wiederhold, G. (1992). Mediators in the Architecture of Future Information Systems, Journal of IEEE Computer, IEEE Computer Society, vol. 25, n°3, Mars 1992, pp. 38-49.

Optimisation combinatoire sur grilles de calcul pour la fouille de données en spectroscopie PIR

Nouredine Melab* Sébastien Cahon* El-Ghazali Talbi*

*CNRS/LIFL and INRIA - University of Lille,
59655 Villeneuve d'Ascq CEDEX, France
{cahon,melab,talbi}@lifl.fr,
http://www.lifl.fr/OPAC/

Résumé. La fouille de données de bases denses et larges se révèle généralement NP-difficile. Différentes approches s'appliquent à l'analyse efficace de telles masses de données. Nous proposons une approche hybride combinant la sélection d'attributs, l'emploi d'heuristiques réduisant ainsi l'espace de décision et enfin la mise en oeuvre du parallélisme à large échelle. L'application traitée relève de la Spectroscopie Proche-Infrarouge (SPI). Elle désigne l'ensemble des techniques appliquées à la prédiction de la concentration de différents constituants d'un échantillon à partir de son absorbance aux radiations lumineuses. Il s'agit d'un problème NP-dur et difficile en pratique, puisqu'associé à une combinatoire importante ainsi qu'un coût CPU à l'évaluation élevé. L'exploitation de la plate-forme de métaheuristiques parallèles ParadisEO a permis la modélisation du problème, la conception d'un algorithme génétique intégrant plusieurs niveaux de parallélisme et de coopération, et son déploiement sur plate-formes de Métacomputing. Les résultats obtenus se montrent convaincants tant dans la performance à l'exécution que dans la qualité des solutions produites. [1]

1 Introduction

La fouille de données est un processus de découverte dans les bases de données (BD) de modèles de prédiction nouveaux, précis, utiles et faciles à comprendre. Ces BD sont souvent denses signifiant que leurs schémas sont constitués d'un grand nombre d'attributs. La prise en compte de tous les attributs pour la construction de modèles de prédiction pose deux problèmes majeurs : des études théoriques et expérimentales Zaki (1999) ont montré que l'exploration des données est de complexité temporelle exponentielle en nombre d'attributs, donc très coûteuse en temps CPU pour des BD denses ; un nombre important d'attributs non pertinent et/ou redondant rendant plus difficile la compréhension des modèles de prédiction extraits. Dans la littérature, différentes définitions de la pertinence ont été proposées. Dans ce travail, nous considérons qu'un attribut est pertinent si son élimination provoque une perte significative de précision du modèle extrait. D'autre part, un attribut est dit redondant s'il est corrélé

[1] Ce travail rentre dans le cadre du projet national ACI Masses de données « Grille Géno Médicale »

avec d'autres attributs. Sa prise en compte dans le modèle n'apporte aucune information supplémentaire aidant à la compréhension de celui-ci.

La sélection d'attributs Liu et Motoda (1998) est une approche permettant de réduire la dimension du problème. Il s'agit de sélectionner un sous-ensemble d'attributs pertinents et non redondants sans dégrader la précision du modèle extrait. Le problème de sélection optimale d'attributs a été montré NP-difficile Narendra et Fukunanga (1977). Plusieurs techniques ont été proposées dans la littérature pour la résolution du problème de sélection d'attributs (Narendra et Fukunanga, 1977; Vafaie et al., 1992; Dash et Liu, 1997; Liu et Motoda, 1998). Ces techniques sont basées sur trois catégories d'approches : les approches « filtrantes » (Almuallim et T.Dietterich, 1991; Liu et Setiono, 1996), les approches « enveloppantes » Kohavi et John (1997), et les approches hybrides Sebban et Nock (2002). Les premières considèrent deux étapes distinctes dans le processus de construction du modèle : d'abord la sélection d'attributs, et ensuite la construction du modèle. Dans ce cas, la sélection n'est pas basée sur la précision du modèle extrait mais sur d'autres critères tels que la corrélation entre attributs. Au contraire, dans les approches enveloppantes, la sélection et la construction du modèle sont mêlées. Le processus global se déroule en plusieurs étapes, et à chaque étape une sélection est effectuée, un modèle prédictif est construit et sa qualité de prédiction est calculée. La qualité de la sélection est améliorée d'étape en étape, et le processus s'arrête lorsqu'aucune amélioration n'est avérée.

Le problème traité ici relève de la sélection d'attributs en spectroscopie proche infrarouge (PIR). Une modélisation de la sélection d'attributs sous forme de problème d'optimisation combinatoire (POC) sera présentée. Notre approche de résolution est basée sur la conjugaison de plusieurs approches, notamment la sélection d'attributs, l'optimisation approchée par les métaheuristiques, et enfin le parallélisme sur grilles de calcul. Nous décrirons l'approche adoptée à la résolution ainsi que sa parallélisation avec la plate-forme ParadisEO dédiée à la résolution parallèle approchée de POCs NP-durs et difficiles en pratique. Les résultats expérimentaux obtenus sont enfin présentés et interprétés.

2 Présentation et données du problème

Le problème d'analyse de spectres auquel nous nous sommes intéressés consiste à construire un modèle de prédiction de la concentration d'un constituant donné dans une matière organique. Comme le montre la figure 1, une source lumineuse envoie des radiations PIR sur différentes longueurs d'ondes vers un échantillon donné. Un spectre d'absorption est ainsi recueilli à l'aide de capteurs sensibles à l'intensité lumineuse. En outre, la concentration du composant étudié dans l'échantillon est mesurée par analyse chimique. L'expérience est renouvelée un certain nombre de fois permettant ainsi d'obtenir les données du problème. Il s'agit d'un lot d'échantillons, chacun constitué de N absorbances (valeurs réelles) et d'une valeur (réelle) de concentration. Pour vérifier la précision du modèle (une fois construit), les échantillons sont répartis en différents lots : un lot de calibrage, un autre de validation et enfin d'un troisième lot de prédiction. Ils sont respectivement dédiés à la construction et la validation du modèle (pendant et après le processus d'optimisation).

Nous avons ici considéré une instance du problème où la matière organique et le constituant désignent respectivement la betterave et le saccharose. Le jeu de données a été constitué par le Laboratoire de spectroscopie et raman de l'Université de Lille1 (LASIR). Les lots de calibrage,

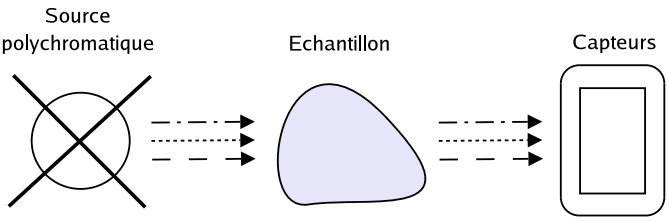

FIG. 1 – *Constitution d'un spectre par mesure de l'absorbance lumineuse sur le domaine PIR.*

de validation et de prédiction contiennent chacun 600 échantillons. Chaque échantillon est constitué de 1020 absorbances (correspondant aux 1020 longueurs d'onde du domaine PIR) et d'une valeur de concentration exacte déterminée par analyse chimique.

Selon la loi de Beer Lambert, la concentration d'un composant est fonction (combinaison linéaire) des absorbances correspondant aux longueurs d'onde qu'il peut absorber. Le problème de prédiction consiste donc à estimer cette fonction (modèle). Comme le problème est linéaire, il est souvent traité avec des méthodes statistiques telles que la méthode des moindres carrés partiels ou (PLS) Wold et al. (1984). La métrique permettant de calculer la précision d'un modèle est l'erreur RMS, obtenue à partir des concentrations prédites (calculées en utilisant le modèle de prédiction) et de celles réelles qui nous sont connues. L'erreur RMS, correspondant à l'écart-type des erreurs de prédiction peut être formulée de la manière suivante :

$$\mathrm{ErreurRMS} = \sqrt{\frac{\sum_{i=1}^{n}(\widehat{c_i} - c_i)^2}{n}}$$

Ici, *n* désigne le nombre d'échantillons du nouveau lot testé. Pour un échantillon *i*, c_i et $\widehat{c_i}$ désignent respectivement ses concentrations réelle et prédite.

3 La plate-forme ParadisEO pour l'optimisation parallèle approchée

En pratique, les POCs se révèlent souvent NP-durs et donc gourmands en ressources CPU. Contrairement aux méthodes exactes (Branch & X), les métaheuristiques (heuristiques génériques) s'appliquent à la résolution « satisfaisante » d'instances de grandes dimensions en un temps raisonnable. On distinguera les métaheuristiques manipulant des populations (e.g. les Algorithmes Génétiques) de celles basées sur des solutions uniques (e.g. la recherche Tabou) favorisant respectivement l'exploration ou l'intensification dans le processus de recherche. L'hybridation de métaheuristiques repose sur l'équilibre entre les phases de diversification et d'intensification de la recherche. Les meilleurs résultats pour de nombreux POCs de grande taille ont été obtenus par des algorithmes hybrides Talbi (2002). Bien que les métaheuristiques réduisent considérablement la complexité temporelle, elles ne s'avèrent plus satisfaisantes de certains problèmes réels pour lesquels le traitement des solutions manipulées (e.g. l'évaluation du coût) se montre important. Le calcul parallèle sur grilles s'est récémment avéré la meilleure solution face à cette problématique Foster et Kesselman (1999).

Dans le cadre du projet national ACI-GRID « Défis en Optimisation Combinatoire sur Grilles » , nous nous attachons à produire des prototypes de logiciels parallèles pour la résolution d'applications en Optimisation Combinatoire de très grande taille. Ces derniers doivent être capables d'exploiter au niveau applicatif, de manière transparente et efficace, les énormes capacités mémoire et calcul offertes par la globalisation des ressources informatiques commes les grilles de calcul. A cet effet, nous avons conçu la plate-forme logicielle ParadisEO [2] Cahon et al. (2004), dédiée à la conception réutilisable de métaheuristiques parallèles hybrides.

Elle intègre l'ensemble des métaheuristiques (algorithmes évolutionnaires et des méthodes à base de recherche locale), la plupart des modèles parallèles Cotta et al. (2005) (basés sur la marche, la solution et la fonction objectif) et des techniques d'hybridation. ParadisEO est basée sur une séparation conceptuelle claire entre la partie invariante liée aux méthodes de résolution et celle spécifique aux problèmes traités, lui conférant ainsi une grande flexibilité et une réutilisation maximale du code et des modèles.

ParadisEO repose sur différents intergiciels ou librairies pour le déploiement sur architectures parallèles et/ou distribuées dédiées. Elle a été récemment étendue afin de permettre l'exploitation efficace de plates-formes de Metacomputing Cahon et al. (2005).

On notera que la gridification sur de tels méta-systèmes accroît la complexité liée à la mise en oeuvre. Il se pose en effet de nouvelles problématiques n'apparaissant pas dans les environnements d'exécution parallèles « traditionnels » . Les aspects suivants ont été identifiés comme apparaissant critiques à l'implémentation : la dynamicité, la volatilité, l'asynchronisme, l'hétérogénéité et enfin le passage à l'échelle. ParadisEO adapte les modèles parallèles pour un déploiement sûr et efficace à l'exécution.

4 Résolution parallèle

L'analyse statistique des données de notre problème a montré une forte corrélation d'une grande partie des longueurs entre elles, et avec la concentration associée. Des courbes illustrant ces mesures de corrélation sont présentées dans Cahon et al. (2005). Ce constat nous a conduit à penser que bien que la méthode PLS soit efficace, sa combinaison avec une sélection d'attributs peut davantage améliorer sa performance.

L'idée est de sélectionner les longueurs d'onde les moins corrélées entre elles et les plus corrélées avec la concentration. Le problème peut être ainsi formulé comme suit : un échantillon de données peut être vu comme un vecteur comprenant les valeurs d'absorbance associées à N longueurs d'onde $(\omega_1,\ \omega_2,\ ...,\ \omega_N)$ et une valeur de concentration réelle c. Si f désigne le modèle alors $c = f(\omega_1,\ \omega_2,\ ...,\ \omega_N)$. Le problème de sélection d'attributs consiste alors à choisir M (tel que $M \leq N$) longueurs d'onde pertinentes et non redondantes parmi les N longueurs d'onde du problème.

L'approche que nous avons proposée est enveloppante combinant un AG avec la méthode PLS. Les individus de la population sont des chaînes binaires de longueur $N = 1020$. Le i^{eme} bit est mis à 1 (resp. à 0) si la i^{eme} longueur d'onde est sélectionnée (resp. n'est pas sélectionnée). A chaque génération, la méthode PLS est appliquée à chaque individu en considérant la sélection d'attributs correspondante. Pour chaque individu, un modèle prédictif est ainsi construit et une erreur de prédiction RMS est calculée. Cette erreur représente la valeur

[2] PARAllel and DIStributed Evolving Objects (`http://www.lifl.fr/OPAC/paradiseo/`)

de qualité (ou *fitness*) de l'individu. Les opérateurs de variation sont les opérateurs classiques i.e. le crossover et la mutation.

Le choix d'une sélection d'attributs s'avère délicat en considérant d'autres critères que la qualité du modèle de prédiction lui-même. L'approche filtrante est donc difficilement envisageable. L'approche enveloppante est plus adaptée à ce type de problème et s'avère plus fiable, cependant celle-ci pose un problème de faisabilité. En effet, le nombre d'attributs est élevé (1020) et donc le nombre de sélections candidates : 2^{1020} l'est également. De plus, le coût d'évaluation de chaque sélection est exorbitant car celle-ci est réalisée par régression PLS, elle-même basée sur un calcul matriciel intensif. L'utilisation du parallélisme à grande échelle s'avère indispensable pour supporter le coût de l'intégration de la méthode PLS en tant que fonction d'évaluation de l'AG.

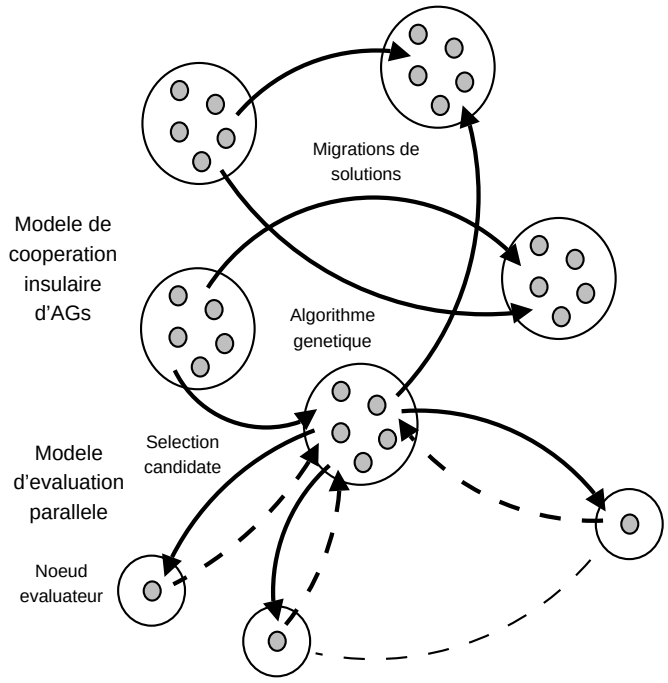

FIG. 2 – *Conception avec ParadisEO d'une métaheuristique intégrant deux niveaux hiérarchiques et complémentaires de parallélisation et d'hybridation*

Le modèle implémenté avec ParadisEO est illustré à la figure 2. Il est constitué d'un modèle coopératif insulaire d'AGs dont la phase d'évaluation est distribuée. Le choix de l'Algorithme Génétique est motivé par sa tendance à l'exploration et à la diversification, à sa capacité à traiter des espaces décisionnels de grandes dimensions. Parmi les différents modèles parallèles de métaheuristiques supportés par ParadisEO, deux niveaux hiérarchiques ont été mis en oeuvre :

– la coopération insulaire reposant sur l'évolution concurrente de populations distribuées dans l'espace. Les AGs hybridés échangent régulièrement des solutions entre elles, se traduisant par une diversification de l'exploration, retardant ainsi le phénomène de

convergence. Il en résulte alors une robustesse accrue et l'obtention de meilleurs résultats.

– L'évaluation parallèle synchrone ou non. En effet, la phase d'évaluation est généralement la plus coûteuse dans l'exécution d'un AG, particulièrement dans le cadre d'applications issues du monde réel. Succédant à la phase dite de transformation, elle consiste à déterminer la qualité de chaque nouvelle solution produite. Ce calcul étant indépendant du reste de la population, la parallélisation est donc ici naturelle et basée sur un partitionnement des individus. Elle ne modifie en rien le comportement de l'AG original, de mêmes résultats sont obtenus plus rapidement.

5 Résultats

L'application conçue avec ParadisEO a ici été déployée et expérimentée sur un environnement de Métacomputing. Le caractère naturellement hétérogène et dynamique des environnements de Métacomputing rend difficile les mesures de performance à l'exécution. Les métriques communément utilisées en calcul haute performance (*e.g.* l'accélération parallèle) ne peuvent convenir ici puisqu'elles ne s'appliquent qu'à un ensemble de ressources homogènes et dédiées. En normalisant le temps CPU nécessaire à la réalisation d'une tâche avec la performance d'un Noeud Travailleur (NT) correspondant, ParadisEO agrège les temps d'exécution et établit des statistiques viables sur différentes exécutions. Ce facteur de normalisation peut être basé sur une information matérielle, si disponible (*e.g.* MIPS, FLOPS, ...). Alternativement, ParadisEO déploie des benchmarks sur les NTs participants afin de quantifier leur performance.

Dans Goux et al. (2000), différentes métriques de performance ont été définies formellement, et tout particulièrement l'efficacité parallèle sur environnements de Métacomputing, composés de ressources de calculs hétérogènes et dynamiques dans l'espace et le temps.

En considérant les informations suivantes,

– $U(i)$, le temps de disponibilité cumulée du NT i,

– $t(j)$, le temps utilisateur passé par le NT $w(j)$ à accomplir la tâche j,

nous pouvons définir l'efficacité parallèle ainsi

$$\mu = \frac{\sum_{j \in \mathcal{J}} t(j)}{\sum_{i \in \mathcal{I}} U(i)}$$

Le tableau 1 établit quelques statistiques mesurées à l'exécution de l'AG. coopératif insulaire parallèle de 16 îles sur un réseau de 122 stations de travail interconnectées dans le cadre universitaire (Polytech-Lille), initialement dédiées à l'enseignement doc non dédiées, et globalement sous-exploitées.

L'efficacité parallèle est d'environ 82% signifiant que l'ensemble des noeuds de calcul volontaires sont exploités de façon efficace. Aussi, dans le calcul de l'efficacité parallèle, le coût lié aux communications apparaît faible en comparaison du temps d'exécution de la fonction objectif. On notera que l'évaluation parallèle asynhrone de la population a ici été mise en oeuvre. Ce modèle, ou l'évaluation est désynchronisée des autres phases de l'AG, se montre adapté à des problèmes dont le coût à l'évaluation est irrégulier ou encore lorsque la plateforme d'exécution est hétérogène et volatile.

Nombre de NTs	122
Durée d'exécution réelle	36953 s. (10 heures)
Durée d'exécution cumulée	2363571 s. (27 jours)
Nombre moyen de NTs actifs	115
Nombre moyen de NTs participants	78
Performance parallèle	0.82

TAB. 1 – *Statistiques mesurées à l'exécution de l'AG. coopératif parallèle*

La figure 3 illustre l'évolution de l'erreur de prédiction correspondant à la meilleure sélection de fréquences au fur et à mesure des générations accomplies pour 1, 4 ou 16 AGs en coopération. On comparera la qualité des modèles prédictifs après sélection préalable des longueurs d'onde ou sans sélection des attributs. La regression PLS sans sélection des longueurs d'onde est associée à une erreur RMS de 0.150. La sélection générée par l'AG. simple permet d'obtenir un gain en précision de 30 %. Dans le modèle hybride, le phénomène de convergence est manifestement retardé. Ainsi, une gain de 32 % est alors mesuré. Remarquons néanmoins que cette forme d'hybridation, si elle offre un apport significatif dans la qualité des solutions produites, a un coût au déploiement très important. En effet, la consommation en ressource CPU est multipliée par le nombre d'AGs coopératifs déployés (*i.e.* 16).

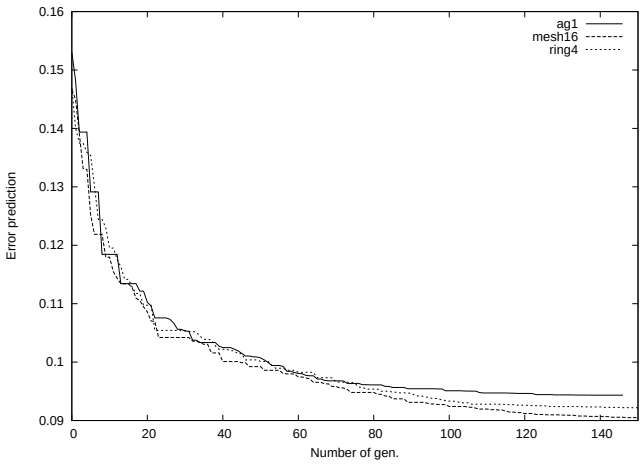

FIG. 3 – *Evolution de l'erreur de prédiction RMS associée à la meilleure solution trouvée avec un modèle coopératif insulaire de 1, 4 et 16 îles.*

La sélection quasi-optimale obtenue se compose d'un ensemble de 122 attributs parmi les 1020 longueurs d'onde candidates initiales. L'élimination de près de 88 % des fréquences permet une accélération très importante du traitement d'analyse de chaque échantillon : 394 μ s. contre 2930 initialement (PC cadencé à 1 Ghz).

6 Conclusion

La sélection d'attributs en Data Mining réduit l'information intiale, en identifiant les attributs redondants ou ceux dont la valeur ne contribuent pas à la phase d'exploitation des données. Les deux principales approches de ce processus, dites filtrante et enveloppante, se distinguent par un coût à la mise en oeuvre et une exactitude antagonistes. De l'approche enveloppante, préférable car adaptée au processus d'analyse des données, émergent deux problématiques majeures : le caractère NP-difficile de ce problème d'optimisation combinatoire et l'évaluation, généralement très gourmande en ressources, d'une sélection candidate. Aussi, l'emploi de méthodes approchées apparaît encore insuffisant. Le développement d'heuristiques doit être accompagné d'un déploiement sur environnements d'exécution parallèles à grande échelle que sont les grilles de calcul. L'exploitation de plate-formes logicielles, telles ParadisEO présentée ici, réduit considérablement l'effort en développement, masquant la forte complexité inhérente à la mise en oeuvre d'applications parallèles sur environnements d'exécution de Métacomputing. Elle a contribué au développement de méthodes efficaces et performantes face à une application industrielle en spectroscopie PIR, difficile en pratique. Les résultats obtenus se montrent très satisfaisants tant en terme de qualité des solutions obtenues que des performances mesurées à l'exécution.

Références

Almuallim, H. et T.Dietterich (1991). Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*.

Cahon, S., N. Melab, et E.-G. Talbi (2004). ParadisEO : A Framework for the Reusable Design of Parallel and Distributed Metaheuristics. *Journal of Heuristics 10*(3), 357–380.

Cahon, S., N. Melab, et E.-G. Talbi (2005). An enabling framework for parallel optimization on the computational grid. In *the Proc. of the 5th IEEE/ACM Intl. Symposium on Cluster Computing and the Grid (CCGRID'2005)*, Cardiff, UK.

Cotta, C., E.-G. Talbi, et E. Alba (2005). Parallel hybrid approaches. In *Parallel metaheuristics*. USA : Edited by E. Alba, J. Wiley and sons.

Dash, M. et H. Liu (1997). Feature selection for classification. *Intelligent Data Analysis 1*(3).

Foster, I. et C. Kesselman (1999). *The Grid : Blueprint for a New Computing Infrastructure*. Morgan-Kaufmann.

Goux, J.-P., S. Kulkarni, M. Yoder, et J. Linderoth (2000). An Enabling Framework for Master-Worker Applications on the Computational Grid. In *HPDC'00 : Proc. of the* 9^{th} *IEEE Intl. Symposium on High Performance Distributed Computing (HPDC'00)*, Washington, DC, USA, pp. 43. IEEE Computer Society.

Kohavi, R. et G. John (1997). Wrappers for Feature Subset Selection. *Artificial Intelligence 97*(1–2), 273–324.

Liu, H. et H. Motoda (1998). *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers.

Liu, H. et R. Setiono (1996). A Probabilistic Approach to Feature Selection - a Filter Approach. In *Proc. of the* 13^{th} *Intl. Conf. on Machine Learning*.

Narendra, P. et K. Fukunanga (1977). A branch and bound algorithm for feature subset selection. In *IEEE Trans. on computer*.

Sebban, M. et R. Nock (2002). A Hybrid Filter/Wrapper Approach of Feature Selection using Information Theory. *Journal of Pattern Recognition 35*(4), 835–846.

Talbi, E.-G. (2002). A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics, Kluwer Academic Publishers Vol.8*, 541–564.

Vafaie, H., J. De, et A. Kenneth (1992). Genetic Algorithms as a Tool for Feature Selection in Machine Learning. In *Proc. of the Intl. Conf. on Tools with AI*, Arlington, VA, pp. 200–204. IEEE Computer Society Press.

Wold, S., A. Ruhe, H. Wold, et W. J. Dunn, III (1984). The collinearity problem in linear regression. the partial least squares (PLS) approach to generalized inverses. *SIAM Journal on Scientific and Statistical Computing 5*(3), 735–743.

Zaki, M. J. (1999). Parallel and distributed data mining : An introduction. In *Large-Scale Parallel Data Mining*, pp. 1–23.

Summary

Mining large and dense databases is generally NP-hard. Different approaches aim at their efficient analysis. We propose a hybrid approach relying on the feature selection, the use of heuristics, hence reducing the search space and then large-scale parallelism (GRID computing) at execution. In this paper, we tackle the Near-Infrared spectroscopy. It denotes the techniques applied to the computation of the concentration of some components for a product from its absorbances to NIR radiations. It is a NP-hard problem in practice, due to both the combinatorial explosion and the CPU cost of the process of evaluation. The framework for parallel metaheuristics ParadisEO enables the modeling of the problem, the design of a Genetic Algorithm embedding several hierarchical layers of parallelism and cooperation and then, its deployment on Metacomputing platforms. Obtained results are convincing both in terms of performane at execution and regards the robustness of the computed solutions.

NDS : Network Distance Service un outil de calcul de distances réseau nœud à nœud

Julien Gossa*, Jean-Marc Pierson*

*LIRIS - INSA Lyon, Bât. Blaise Pascal, 69 621 Villeurbanne, France
prenom.nom@insa-lyon.fr,
http://liris.cnrs.fr/j̃gossa/nds

Résumé. Nous présentons dans cet article un système permettant d'évaluer les *distances* hôte à hôte sur un réseau étendu tel qu'une grille et ce de façon flexible, c.a.d. adaptée à la tâche envisagée et au contexte d'utilisation. Nous appelons *distance* une valeur réelle représentant l'efficacité d'un couple d'hôtes lors d'une interaction particulière. Elle constitue une aide dans la sélection parmi différents hôtes pouvant fournir le même service. Nous détailons notre proposition et la modélisation des différentes données utilisées ainsi que l'interface du Service Web que nous avons développé sous Globus Toolkit 4.

1 Introduction

A l'heure actuelle, les infrastructures de grilles sont en passe de devenir efficaces d'un point de vue fonctionnel. En effet des intergiciels tels que globus (Foster et Kesselman (2004)), profitant des efforts récents de standardisations relatifs aux services web, permettent la mise en place d'une grille comportant (plus ou moins bien) toutes les caractéristiques requises : référencement des ressources, distribution de tâches, gestion des données disponibles et répliquées, etc... Les efforts actuels sont encore concentrés sous le point de vue fonctionnel avec le développement des architectures et des protocoles nécessaires. Un des domaines d'application de notre service est la gestion de données largement dissiminées. Une grille peut aujourd'hui supporter un très grand nombre de données distribuées. Ces dernières peuvent être de tailles très variées, et souvent conséquentes, et peuvent être répliquées à échelles très variées, et potentiellement très grandes. A ce jour, les grilles permettent de stocker et référencer ces données et leurs copies (voir RLS dans Foster et Kesselman (2004)), mais aucun outil ne permet encore d'effectuer une sélection de façon uniforme, simple et flexible parmi toutes ces copies. Ce choix est pour l'instant laissé à la discrétion de l'utilisateur qui est contraint, la plupart du temps, à se limiter à une sélection hasardeuse ou à développer lui même un système spécifique.

Or, la réplication des données est une réponse à de nombreuses problématiques : disponibilité, persistance, performance... La phase de sélection parmi plusieurs copies d'une même donnée est particulièrement critique. Non seulement pour son consommateur puisqu'elle conditionne le temps de rapatriement et le taux d'échec (copie supprimée, ressource indisponible,...), mais aussi pour le fonctionnement du réseau en lui-même puisqu'elle conditionne la saturation ou la sous exploitation de ses ressources (de stockage et communication).

La suite du document est organisée comme suit : nous présentons l'état de l'art en section 2, puis les détails de notre proposition en section 3. Nous la discutons en section 4 et concluons en section 5.

2 Etat de l'Art

Les choix de sélection sont intimement liés à l'aspect monitoring. Plus particulièrement la gestion des *métriques* est un domaine de recherche très actif actuellement. Cette notion peut être assez difficile à appréhender, comme l'indique Lowekamp et al. (2004), mais nous pouvons la résumer intuitivement comme "des caractéristiques de l'état courant des différentes ressources d'un réseau". Alors que certains travaux tels que Lowekamp et al. (2004) et Ziviani et Schulze s'intéressent au problème plus fondamental de l'identification des métriques appropriées à certains environnements et leur différents moyens d'observations, d'autres comme Wang et Zhou et Leese et al. (2005) s'intéressent au problème plus architectural de la collecte des différentes observations pour les intégrer dans des catalogues et les rendre disponibles aux applications.

Le fait est que les catalogues de ressources de grille actuels (voir MDS dans Foster et Kesselman (2004)) intègrent assez bien les métriques relatives aux hôtes, mais très mal les métriques relatives aux communications. En effet, les services de monitoring s'intéressant aux machines et aux communications, tels que Leese et al. (2005) et Obertelli et Wolski, sont peu ou pas intégrés à l'intergiciel proprement dit, et restent donc au stade d'outil additionnel. Or, cette intégration est d'une réelle importance puisqu'elle est indispensable à l'utilisation des métriques pour optimiser le fonctionnement de la grille dans son activité quotidienne et à tous les niveaux de son fonctionnement.

Une autre observation est que nous n'avons pu trouver qu'un seul travail (Ferrari et Giacomini (2004)) proposant une méthode de combinaison de différentes métriques afin de comparer différents réplicas d'une même donnée à rapatrier. Les auteurs exposent une fonction dite de "proximité" combinant le Round Trip Time (RTT) maximum et actuel avec le débit maximum et actuel d'un lien pour évaluer la distance entre deux nœuds. Nous pouvons faire trois observations :

- La mise en œuvre de cette combinaison reste à définir.
- Une méthode générique permettant de déclarer de nouvelles combinaisons est nécessaire.
- Les caractéristiques de la tâche envisagée ne sont pas pris en compte dans le calcul de cette "proximité" : elle sera la même que l'on veuille rapatrier un fichier de 10 Go, stocker un fichier de 1 ko ou encore soumettre une tâche.

3 Proposition

Notre proposition concerne précisément ces trois derniers points. Pour l'illustrer, nous prenons le problème dit du « pas niais de la ménagère ». Lorsque le voyageur de commerce doit acheter du pain, il va dans la boulangerie qui minimise la distance kilométrique à parcourir (illustrant ce qui se fait à l'heure actuelle en matière de sélection de ressources sur l'Internet

où l'on se limite bien souvent à une seule métrique uniforme telle que le nombre de hops). Par contre, la ménagère, pas niaise, prend en compte beaucoup plus de paramètres : le prix et la qualité du pain, la qualité de l'accueil, les horaires de fournées... mais aussi le temps d'attente prévisible en fonction du jour, de l'heure et du nombre de vendeurs ou encore son encombrement et le détour qu'il faudra faire en fonction des autres achats qu'elle prévoit de faire. Notre outil lui permet de calculer une distance subjective à chacune des boulangeries en fonction du contexte actuel (jour, heure, attente prévisible...) et de son intention (qualité du pain, temps d'attente maximum, planning de la journée...).

Plus concretement, notre outil permet de calculer une *distance* entre des hôtes clients potentiels et chacun des hôtes fournisseurs potentiels, relativement au service attendu et à l'état courant des ressources. Plus la *distance* entre un couple client/fournisseur est faible, plus ce couple est propice à rendre le service dans les meilleures conditions. Cette *distance* est une valeur réelle non soumise aux propriétés classiques des distances mathématiques, ce terme doit donc être compris de façon intuitive. Notre outil a donc pour but d'aider son utilisateur dans le processus de sélection du (ou des) meilleur(s) hôte(s) parmi les nombreuses possibilités qui s'offrent à lui. Par exemple, il permettra de sélectionner parmi plusieurs copies d'une donnée celle qui est la plus appropriée au rapatriement sur un certain hôte en fonction de l'état des latences et bandes passantes, ainsi que des caractéristiques de cette donnée (sa taille, si elle supporte l'accès par flux, etc...). Sous un point de vue logique, notre outil permet de rajouter aux catalogues de grilles (catalogue d'hôtes - MDS, catalogue de réplicas - RLS, ou autre...) les informations de pertinence relatives au contexte et à la tâche envisagée.

Notre optique n'est pas de nous arrêter à la sélection d'une copie de donnée pour le rapatriement, mais de l'étendre à toutes les tâches possibles où cette notion de distance peut s'avérer utile : sélection d'un lieu de stockage, soumission de tâche, ordonnancement...

3.1 Modélisation des données

3.1.1 Côté serveur : les métriques et leur observations

Nous appelons métrique une caractéristique mesurable d'une ressource disponible sur le réseau, que ce soit un hôte ou bien une route.
Une métrique m est un 3-tuple $< name, description, commandLine >$, où :

- $m.name$ est l'identifiant unique de la métrique utilisé au sein de NDS.
- $m.description$ est le description littérale fournie au client sur demande spécifiant en particulier le type des observations, l'unité utilisée et les différents paramètres à stipuler.
- $m.commandLine$ est la ligne de commande exécutée par le serveur pour obtenir la valeur de l'observation de m.

Nous désignons pas M l'ensemble des métriques m supportées par le serveur.

Nous appelons mesure ou observation, la valeur utile correspondant à une métrique à un instant donné. Cette mesure est effectuée par des outils externes appelés par $m.commandLine$. Notre outil reste ainsi très facilement extensible et ouvert à tous les outils imaginables. Nous préconisons l'utilisation de MDS pour les métriques relatives aux hôtes et de NWS pour les métriques relatives au réseau. Toutes les mesures doivent cependant pouvoir être disponibles

depuis le serveur NDS.

Une observation o est un 3-tuple $< name, type, value >$, où :
- $o.name = m.name$ tel que $m \in M$.
- $o.type$ est le type de la mesure, $o.type \in \{int, double, String\}$.
- $o.value$ est la valeur de la mesure, résultat de $m.commandLine$.

Il est important de noter que nous prenons des libertés vis-à-vis du terme “métrique”. En effet une même métrique peut avoir de nombreuses façons d’être mesurée qui vont donner autant de résultats différents. Ici nous confondons métrique et moyen de mesure, si bien que deux moyens de mesure différents d’une même métrique (par exemple avec ou sans prévision) correspondront en fait à deux éléments distincts de M.

3.1.2 Côte client : les description d’intentions et la fonction distance

Nous appelons intention le cadre dans lequel les mesures seront utilisées et IntentionDescription (ID) l’ensemble des caractéristiques définissant ce cadre. Une caractéristique c est une valeur nommée et typée sans restriction particulière : $c =< name, type, value >$.
Une ID est un ensemble non ordonné de n caractéristiques : $id = \{c_1, c_2, ..., c_n\}$, $n \in [0, \infty]$.
Le client se doit de rester cohérent puisque les unités ne sont ni spécifiées ni imposées et que les valeurs effectives ne sont pas dissociées de la déclaration des différents champs.

Nous appelons fonction distance, ou DistanceFunction (DF), la fonction relative à une ID particulière permettant de combiner les observations disponibles et les valeurs de l’ID pour finalement obtenir la distance entre deux hôtes. Cette fonction doit retourner un double et peut supporter les comparaisons booléennes ainsi que les traitements de chaînes de caractères. Nous nous sommes basé sur JEP (JEP), la syntaxe doit donc être conforme à ses spécifications. L’utilisation des valeurs des IDs et des métriques est rendue transparente dans cette fonction. Pour utiliser une caractéristique c, il suffit d’y inclure une variable $c.name$. Pour utiliser la métrique m, il suffit d’y inclure une fonction $m.name()$ dont les paramètres sont stipulés dans $m.description$. Les mots clés $\%From$ et $\%To$ remplacent les IP ou URI des machines sources et destinations stipulées lors de l’invocation du service.

3.2 Les fonctions du Service Web

Notre service, développé sous GT4 (Foster et Kesselman (2004)) est composé de deux fonctions essentielles :
- $getMetricsDescription$ qui renvoie M.
- $getDistances$ qui prend en paramètre des ensembles de sources/destinations, une ID et la DF associée et renvoie une distance pour chaque couple source/destination.

Un client devra donc invoquer une première fois le service via la fonction $getMetricsDescription$ puis utiliser ces données pour construire son ID et sa DF. Il pourra ensuite invoquer $getDistances$ pour chaque prise de décision.
La fonction principale de $getDistances$ est de remplacer dans DF les $id.name$ par $id.value$, ainsi que les $m.name()$ par $o.value$ après avoir appelé $m.commandLine$ avec les les IPs/URI effectifs des couples sources/destinations à la palce des $\%From$ et $\%To$. Elle renvoie ensuite une évaluation de la DF modifiée pour chaque couple source/destination.

3.3 Un exemple concret très simple : le raptriement d'une donnée

L'ensemble des métriques utilisées fournies par le serveur sont :
$M = \{< NWSlatencyTcp$,"NWS latency from param1 to param2", ... $>$,
$< NWSbandwidthTcp$,"NWS bandwidth from param1 to param2", ... $>\}$
Le client définit ensuite :
$DataGetID = \{< data\_size, int, x >\}$.
$DataGetDF = (NWSLatency(\%From, \%To) + NWSLatency(\%To, \%From))/1000$
$+ data\_size/(NWSbandwidthTcp(\%To, \%From) * 128)$

Lors de l'invocation du service, x est remplacé par la taille de la donnée en octets.
Pour plus de lisibilité, nous noterons $DataGetCF = RTT + x/BP$. 1000 permet de convertir le RTT en s et 128 de convertir la BP en o/s.
Avec, les valeurs moyennes de BP de l'ordre de 100 kb/s et des valeurs de RTT de l'ordre de 100 Ms observées sur l'Internet, on obtient $DataGetCF \approx .1 + x/12800$. On peut alors observer que si $x << 1280$ la latence sera privilégiée face à la bande passante, ce qui est le plus pertinent avec des données de faible taille. Avec $x >> 1280$, ce sera la bande passante qui sera privilégiée, ce qui reste pertinent avec des données de grande taille.
Il est à noter que ce seuil de 1280 octets (qui correspond à la taille des données utiles transportées par un paquet TCP/IP) dépend grandement du contexte : RTT et BP moyennes, MTU, protocole de rapatriement, etc... Tous ces paramètres peuvent être intégrés dans des ID et DF plus complexes et rester transparentes à l'utilisation.
Des exemples plus complexes sont disponibles disponibles sur le site du projet.

4 Discussion et Perspectives

Ce type d'outil peut très facilement aller à l'encontre du concept fondamental de transparence cher aux grilles : l'utilisateur devrait choisir lui même à quel hôte s'adresser et donc avoir une connaissance approfondie de la topologie de la grille. En fait, nous désignons par "utilisateur" de notre service, un administrateur ou un développeur de la grille qui aurait besoin de rajouter de la pertinence dans son application. Un exemple est le projet GGM (Pierson et al. (2005)) où notre service intervient dans : le service d'optimisation de requêtes, l'entrepôt de données distribué et le service de cache distribué. De plus, il est une brique de base du service de placement optimisé des données FReDi (Gossa et al. (2006)).
Notre service étant exclusivement un service de consultation et les communications nécessaires très limitées si il est exécuté sur la même machine que les outils de mesures sous-jacents, le problème du passage à l'échelle ne se pose pas.
Etant basé sur les Services Web, il profite de leurs avantages mais en récupère également leurs inconvénients, en particulier le manque de légèreté. Ce service peut en effet être rendu de façon beaucoup plus rapide avec un développement spécifique, ce que nous recommandons pour un projet en faisant une utilisation intensive.
Nos perspectives sont l'évaluation de notre solution pour la sélection de réplicas de données à rapatrier et à stocker avec des données aux caractéristiques variées et des états de ressources variés. Son utilisation n'est pas restreinte au cadre de la gestion de données, mais peut s'appliquer à de nombreux domaines touchant au distribué. Ce service sera en particulier utilisé

dans un système de cartographie de grille qui permettra, sous les contraintes fixées par son utilisateur, de visualiser une topologie flexible et adaptée de son environnement de travail.

5 Conclusion

Nous avons présenté dans cet article un système dŠévaluation de la collaboration entre deux hôtes dŠune grille. Cette évaluation est résumée sous la forme dŠune valeur décimale appelée distance qui permet de sélectionner les meilleurs hôtes dans un contexte particulier. Notre système est très facilement extensible et intégrable dans toute architecture ou simple service qui en ressentirait le besoin. Il représente un véritable gain de temps et une aide essentielle dans lŠoptimisation de tout service de grille confronté à des choix de ressource, donc essentiellement en présence de réplication de données ou de services.

Références

Jep - java math expression parser. www.singularsys.com/jep/.

Ferrari, T. et F. Giacomini (2004). Network monitoring for grid performance optimization. *Computer Communications 27*(14), 1357–1363.

Foster, I. et C. Kesselman (2004). *The Grid : Blueprint for a New Computing Infrastructure*. Morgan Kaufmann.

Gossa, J., J.-M. Pierson, et L. Brunie (2006). Fredi : Flexible replica displacer. *ICN'06 - à paraitre*.

Leese, M., R. Tyer, et R. Tasker (2005). Network performance monitoring for the grid. UK e-Science.

Lowekamp, B., B. Tierney, L. Cottrell, R. Hughes-Jones, T. Kielmann, et M. Swany. (2004). A hierarchy of network performance characteristics for grid applications and services. *Global Grid Forum*.

Obertelli, G. et R. Wolski. Network weather service. http ://nws.cs.ucsb.edu/.

Pierson, J.-M., L. Brunie, M. Miquel, A. Tchounikine, C. Dhaenens, N. Melab, T. E. ghazali, A. Hameurlain, et F. Morvan (2005). Grid for geno-medicine. *BioGrid'05*.

Wang, J. et M. Zhou. Provinding network monitoring service for grid computing. *FTDCS'04*.

Ziviani, A. et B. Schulze. Combining grid computing and internet measurements. *WCGA'05*.

Summary

We present a flexible network end-to-end distance evaluation system designed to work on large scale networks such as grids. We call *distance* a real value representing the effectiveness of a hosts couple for a particular interaction. This value is adapted to the envisaged tasks' characteristics and its execution context. It helps the selection among different hosts running the same service. We detail our proposition, the data modeling and the interface of our WebService developed within Globus Toolkit 4.

Gestion de la Sécurité pour l'Extraction Parallèle Distribuée des Connaissances

Serge Chaumette*, Achraf Karray*,** et Damien Sauveron***

*LaBRI, UMR CNRS 5800 – Université Bordeaux 1
351 cours de la Libération, 33405 Talence CEDEX, FRANCE.
serge.chaumette@labri.fr
**ENIS – Université de Sfax
BP W.3038 Sfax, TUNISIE
achraf.karray@labri.fr
***XLIM, UMR CNRS 6172 – Université de Limoges
83 rue d'Isle, 87000 Limoges, FRANCE.
damien.sauveron@unilim.fr

Résumé. Dans le cadre de l'extraction de connaissances où les données et le code de fouille appartiennent à des propriétaires différents et qui ne se font pas nécessairement confiance, il semble difficile de concilier la confidentialité et l'intégrité des données d'une part et celles du code et de son exécution d'autre part. C'est une solution à ce problème non trivial que nous apportons grâce à l'utilisation d'entrepôts de données distribués sur la grille de cartes à puce que nous avons mise en place. Dans ce document nous décrivons l'exploitation de nos travaux dans le cadre d'une application particulière.

1 Introduction

Les grilles de calcul ou les architectures de type grappes ou clusters de stations sont aujourd'hui largement utilisées pour la fouille de données. Néanmoins, aucun système n'offre actuellement des garanties fortes de sécurité. Les objectifs que nous poursuivons sont : d'assurer la confidentialité et l'intégrité des données traitées ; de garantir la confidentialité et l'intégrité des algorithmes ou critères de fouille utilisés et de l'exécution de ces algorithmes. Pour assurer ces contraintes fortes de sécurité nous utilisons la grille de Java Cards™[1] (présentée Fig. 1) que nous avons mise en place dans le cadre de nos travaux sur la sécurité des systèmes distribués. Nous avons déjà déployé sur cette grille des applications sécurisées dans des domaines variés tels que la sécurité du calcul distribué, la sécurité d'une infrastructure PKI et maintenant la sécurité pour la fouille de données.

Nous aborderons dans la première section la problématique de la sécurité du code et des données dans le cadre de la fouille de données. Dans la seconde section, nous présenterons

[1] Java and all Java-based marks are trademarks or registered trademarks of Sun microsystems, Inc. in the United States and other countries. The authors are independent of Sun microsystems, Inc. All other marks are the property of their respective owners.

FIG. 1 – *La grille de Java Cards.*

la grille de Java Cards (Chaumette et al., 2003; Chaumette et Sauveron, 2003; Chaumette et al., 2005; Atallah et al., 2005; Karray, 2004), son architecture matérielle, logicielle, et les outils d'administration actuellement disponibles. Nous exposerons dans la troisième section l'utilisation que nous faisons de cette grille dans le cadre d'une application spécifique que nous avons développée. Enfin nous présenterons brièvement les travaux en cours et nous conclurons.

2 Confidentialité et intégrité pour la fouille de données : impossible défi ?

Le contexte dans lequel nous nous plaçons est celui où le propriétaire des données et le propriétaire du code qui opère la fouille sont distincts. C'est évidemment un cas fréquent dans le domaine de l'extraction de connaissances. Bien souvent dans ce cadre, on souhaiterait pouvoir s'assurer que le jeu de données et le code de fouille ainsi que son exécution respectent certaines propriétés de sécurité telles que la confidentialité et l'intégrité. Toutefois, de telles contraintes posent beaucoup de problèmes puisqu'aucune des parties ne veut ni dévoiler ni mettre en péril ses biens (les données ou le code et son exécution). Ainsi aujourd'hui, faute de solution, soit les applications ne sont pas mises œuvre, soit les exigences de sécurité sont diminuées. Dans la section 4 nous illustrerons ce propos sur une application et nous montrerons comment à l'aide de notre grille de Java Cards nous pouvons assurer les propriétés de confidentialité et d'intégrité du jeu de données, du code et de son exécution. Toutefois, nous allons tout d'abord examiner comment sont classiquement assurées les propriétés que nous avons énoncées.

2.1 Confidentialité et intégrité du jeu de données

Une solution classique lorsqu'on a un jeu de données que l'on veut partager et dont on veut assurer certaines propriétés de sécurité consiste à définir une partie privée inaccessible (*i.e.* qui restera confidentielle et intègre) et une partie publique qui sera en libre accès pour une simple consultation (*i.e.* pour assurer au minimum l'intégrité) ou en modification (*i.e.* pour assurer au minimum la confidentialité) ou en consultation et en modification (*i.e.* si on

ne souhaite assurer ni la confidentialité ni l'intégrité). On songera bien évidemment à assurer l'authenticité (ce n'est qu'une forme particulière de l'intégrité qui permet de s'assurer qu'on manipule les données originales) dans les cas de figure où cela est nécessaire, par exemple quand la responsabilité des fournisseurs de données risquerait d'être engagée. Bien souvent, l'implémentation de ces mécanismes se fait au travers d'une API publique qui offre un accès contrôlé aux diverses informations de la partie publique. Si le code de fouille de données est installé sur la machine partageant et hébergeant les données on utilise généralement un mécanisme de *sandbox* afin qu'il ne tente pas d'outrepasser ses droits par un accès direct au dépôt de données. On peut aussi envisager d'auditer soit même le code ou de le faire auditer par un tiers de confiance pour s'assurer de son respect de l'API et de son innocuité pour le reste des données confidentielles. Si le code de fouille accède à l'API via le réseau (*e.g.* pour l'interrogation d'une base de données) il est seulement nécessaire de garantir l'intégrité des messages échangés et l'authenticité des données reçues par le programme de fouille.

2.2 Confidentialité et intégrité du code et de son exécution

Les critères de fouille doivent rester secrets afin d'éviter par exemple que le fournisseur de données ne modifie ces dernières pour tromper le programme de recherche. On remarque d'ailleurs que la politique de confidentialité du code de fouille est en contradiction avec la solution proposée auparavant et consistant à permettre l'analyse du code pour s'assurer de son innocuité. De plus, pour assurer la viabilité de la fouille, il faut s'assurer que le code ne peut être modifié (*i.e.* garantir son intégrité) sans quoi les réponses ne seraient pas fiables. De la même façon il faut être capable de garantir la confidentialité de l'exécution et son intégrité.

En effet dans le cas où le code de fouille de données est installé sur une machine n'appartenant pas au propriétaire de ce code, il est facile de faire la rétro-conception du programme pour en analyser le contenu et donc de découvrir les critères de fouille. Il est aussi très facile de modifier ce code binaire pour perturber la recherche. De façon très simple, on peut aussi tracer l'exécution et donc connaître le code et les critères employés (*e.g.* l'administrateur peut faire une copie de la mémoire). Enfin il est facile de modifier l'exécution du programme, par exemple en utilisant des perturbations lumineuses (Govindavajhala et Appel, 2003).

2.3 Vers une solution : le tiers de confiance

Actuellement la seule véritable solution consiste à faire un compromis via un intermédiaire commun auquel les deux parties font confiance et qui assure la protection des données, du code et de son exécution.

Nous utilisons pour notre part une solution similaire mais notre intermédiaire est la carte à puce ou une émanation plus performante et aussi sécurisée. En effet les cartes à puce et plus particulièrement les Java Cards[2] (Sun microsystems, 2003; Chen, 2000) peuvent réellement permettre de résoudre ce défi grâce :

– au fait qu'elles sont évaluées par un CESTI (Centre d'Évaluation de la Sécurité des Technologies de l'Information) et certifiées en France par la DCSSI (Direction Centrale

[2] Java Card est une technologie permettant de faire fonctionner des programmes écrits en Java sur des cartes à puce. Cette technologie définit une plate-forme multi-applicative, portable et sécurisée pour les cartes à puce.

de la Sécurité des Systèmes d'Information) ce qui assure qu'on peut avoir une grande confiance dans les mécanismes sécuritaires qu'elles mettent en œuvre ;
- aux mécanismes sécurisés d'isolation des applications et de partages de données ;
- aux protections physiques offertes (*e.g.* blindage électromagnétique pour bloquer les émanations, pompe de charge pour lisser la consommation en courant, détecteur de conditions anormales de fonctionnement) qui protègent le code chargé et son exécution.

3 La plate-forme Java Card Grid

L'objectif du projet Java Card Grid consiste à définir et à mettre en œuvre une plate-forme matérielle et logicielle ainsi que les outils d'administration associés pour mener des expérimentations sur des applications ayant de fortes contraintes de sécurité.

3.1 La plate-forme matérielle

La plate-forme matérielle que nous avons déployée est représentée Fig. 2.

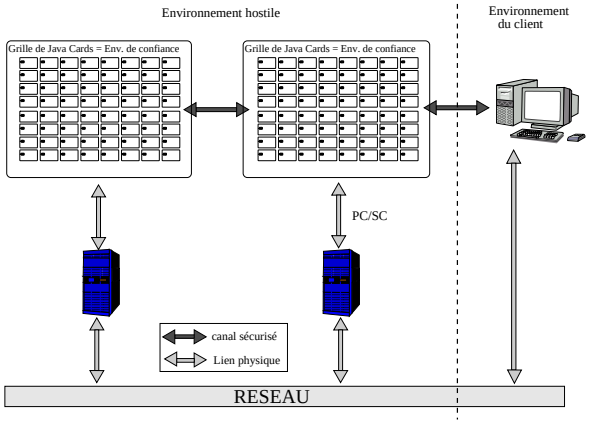

FIG. 2 – *La plate-forme matérielle de la grille de Java Cards.*

En pratique, nous avons déployé deux grilles interconnectées ensemble par un réseau filaire classique. Le matériel est assemblé dans une tour rackable d'une capacité de 19 unités. Chaque grille est composée de :
- un PC occupant 2U ;
- deux racks de SmartMount occupant 2U et comportant chacun 8 lecteurs de cartes de SCM Microsystems respectant le standard CCID, soit un total de 16 lecteurs CCID ;
- trois hubs USB de 7-ports (placés dans un rack vide de 2U) pour alimenter les lecteurs et les connecter au PC ;
- des Java Cards de différents fabricants insérées dans les lecteurs.

Nous avons aussi équipé un des PC d'un écran LCD et d'un clavier rackable spécial pour serveur (avec un touchpad intégré) que nous utilisons pour contrôler les machines. Une photo de la plate-forme est présentée Fig. 1.

3.2 L'infrastructure logicielle

L'infrastructure logicielle que nous avons conçue et implémentée comprend quant à elle deux couches : une couche de bas niveau qui gère les PC, les lecteurs et les cartes ; une couche de haut niveau qui gère l'infrastructure de calcul distribué que nous proposons.

À bas niveau, les PCs de contrôle tournent sous Linux et nous utilisons `pcsc-lite` (Corcoran et al.), une implémentation *open source* du standard PC/SC[3], pour gérer les lecteurs. Pour les lecteurs CCID[4] nous utilisons l'implémentation d'un pilote générique *open source* pour `pcsc-lite` (Rousseau). Par ailleurs comme nous voulions utiliser pour notre infrastructure une couche de haut niveau écrite en Java, nous avons choisi d'employer JPC/SC (IBM BlueZ Secure Systems), un wrapper JNI pour PC/SC, pour contrôler les lecteurs et gérer les applications embarquées dans les cartes (*cf.* Fig. 3).

Au niveau utilisateur, l'API de programmation se base sur Mandala (Chaumette et Vignéras, 2003). Mandala est un *framework* général qui a été développé dans notre équipe pour supporter le calcul distribué en Java. Il fournit une abstraction à la RMI mais avec un mécanisme supplémentaire d'invocation asynchrone de méthode (Vignéras et Grange) qui se révèle très utile dans un environnement où les ressources de calcul sont lentes.

L'ensemble de l'infrastructure logicielle déployée sur les deux machines est présentée Fig. 3.

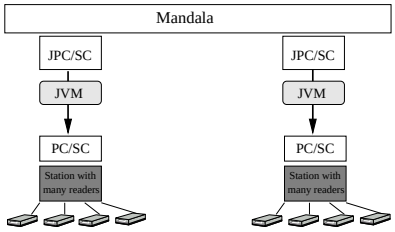

FIG. 3 – *La plate-forme logicielle de la grille de Java Cards*

3.3 Outils d'administration

Nous avons débuté la conception et l'implémentation de différents outils pour administrer la grille de Java Cards, *i.e.* pour surveiller et contrôler la topologie de la grille, pour détecter les cartes défectueuses, pour déployer de nouvelles applications, etc. Pour l'instant, seul l'outil de surveillance à distance de la topologie est disponible.

[3]PC/SC (PC/SC Workgroup) est un standard qui fournit une API de haut niveau pour communiquer avec les lecteurs de cartes à puce.

[4]CCID est un standard qui définit un protocole permettant à tout lecteur compatible CCID de dialoguer avec un hôte sans qu'il soit nécessaire d'installer un pilote spécifique.

4 Une application réelle : CBP/Air France

L'objectif de l'application que nous allons présenter est d'illustrer comment nous pouvons assurer la confidentialité et l'intégrité des données et du code (mais aussi de son exécution) quand ceux-ci appartiennent à deux entités distinctes ne se faisant pas confiance. Plus précisément l'application porte ici sur la recherche d'information dans les fichiers passagers d'une compagnie aérienne, par exemple Air France, par une agence gouvernementale, par exemple le bureau des douanes et de la protection des frontières des États-Unis, le CBP. Cette application correspond à des préoccupations réelles que nous décrivons ci-après.

4.1 Présentation du problème

Depuis le 11 septembre 2001, pour des raisons de sécurité, le CBP demande les informations (nom, numéro de carte de crédit, numéro de téléphone, nombre de bagages, etc.) contenues dans les *passenger name records* (PNR) (Transportation Security Administration), les fichiers de réservation des compagnies aériennes pour les vols à destination des États-Unis, afin de pouvoir effectuer certaines recherches. Selon Washington, ces données peuvent servir à identifier d'éventuels terroristes avant leur entrée sur le territoire américain. Le problème est que si une compagnie aérienne souhaitait préserver la confidentialité des données de ses passagers et donc ne pas communiquer son fichier au CBP, celui-ci pourrait annuler l'autorisation de la compagnie d'atterrir sur le territoire américain, mettant un terme à son activité commerciale vers les États-Unis.

Par ailleurs, dans le cas des compagnies aériennes opérant en Europe, les lois en vigueur dans la Communauté Européenne ont été récemment renforcées par des directives visant à contraindre ces compagnies à préserver la confidentialité des informations personnelles des passagers qu'elles transportent.

Ainsi, on se retrouve dans une situation assez étrange où les compagnies doivent à la fois préserver l'accès aux fichiers de passagers, comme la loi le leur impose, mais également les partager avec le CBP si elles veulent continuer leurs activités vers les États Unis. Pour légaliser cette situation ubuesque qui dans les faits a débuté en décembre 2003, le Conseil des ministres européen a entériné un accord de transfert avec les États-Unis en mai 2004 et ce contre l'avis du Parlement européen qui estimait que ce compromis représentait « une violation des droits fondamentaux » des citoyen européens (Parlement Européen, 2004). D'ailleurs les eurodéputés ont saisi pour arbitrage la Cour européenne de justice. En novembre 2005, l'avocat général indépendant nommé par la Cour européenne de justice a remis ses conclusions demandant l'annulation de la décision. Il s'agit maintenant d'attendre que les juges de la Cour se prononcent car ils ne sont pas tenus de suivre l'avis de l'avocat général (Cour de Justice des Communautés Européennes, 2005). De toutes façons quand le verdict définitif sera rendu l'accord sera quasi obsolète, puisque valable pour 3 ans (2004-2007). Afin de proposer une solution technique viable à cet imbroglio juridique nous avons mis en place une application tirant partie de la plate-forme de confiance que constitue la grille de Java Cards.

4.2 L'échec des solutions classiques

Une solution pourrait être que le CBP fournisse à Air France son application de fouille de données. Air France pourrait ainsi lancer cette application sur les fichiers de passagers

et ne retourner au CBP que la liste des passagers potentiellement suspects. Mais plusieurs problèmes se poseraient car comme nous l'avons déjà souligné le CBP ne souhaite pas dévoiler son algorithme, *e.g.* les critères discriminants mis en œuvre dans son analyse des données des passagers. Or, une analyse de rétro-conception du programme permettrait de les découvrir. Par ailleurs, le CBP ne souhaite pas non plus voir l'exécution de son code espionnée ou modifiée. Donc le code devrait être exécuté soit sur des machines du CBP soit sur une plate-forme de confiance. Enfin, si la compagnie aérienne acceptait de faire tourner le programme du CBP chez elle, elle ne pourrait pas être sûre que ce dernier n'a pas placé de code malicieux dans le programme afin d'avoir accès à toutes les données. Aussi, si la compagnie acceptait de faire tourner le code, cela pourrait seulement être sur une plate-forme de confiance dans laquelle le code ne pourrait pas faire plus que ce que le propriétaire de cette plate-forme lui a permis.

Il se dégage donc à l'évidence un fort besoin de disposer d'une plate-forme de confiance sur laquelle sera déployée le fichier de passagers et l'exécution du code de fouille de données du CBP.

4.3 Notre application

L'objectif de notre application est de permettre au CBP de rechercher des passagers suspects dans le fichier d'Air France, tout en garantissant les deux points suivants : le CBP ne doit pas pouvoir directement identifier les passagers ; Air France ne doit pas avoir connaissance des critères utilisés par le CBP. Pour ce faire, le fichier passagers d'Air France est déployé sur les processeurs (*i.e.* les cartes) accompagné d'une interface (API) permettant d'accéder à tout ou partie des informations contenues sans en violer la confidentialité. En particulier, un passager est identifié par une clef indépendante de son identité effective. Le code du CBP est lui aussi déployé sur les cartes ; il réalise une interrogation et centralise les clefs des passagers considérés comme suspects sur une carte particulière. Air France peut ensuite analyser ces clefs, effectuer une enquête plus poussée sur tel ou tel passager, et le cas échéant, donner les identités des personnes concernées au CBP.

4.4 Implémentation

Dans le cadre de notre application nous avons développé une interface graphique permettant à la compagnie Air France de saisir la liste de ses passagers et de la déployer sur les cartes. Dans cette application, un passager est représenté sur les cartes par la classe `Passenger` contenant toutes les informations le concernant, notamment son nom, son prénom, sa nationalité, etc. Sur chaque carte, nous disposons également d'une *applet* appelée `PassengerManager` qui permet à Air France de gérer la liste de ses passagers.

Cette *applet* `PassengerManager` fournit une interface réduite permettant au CBP de récupérer les informations spécifiées publiques par Air France. Pour cela, la classe `PassengerManager` implante une interface `PassengerInterface` (qui hérite de l'interface `Shareable`) qui donne seulement un accès partiel et contrôlé aux fichiers de passagers et ne permet pas d'identifier un passager particulier (*cf.* Fig. 4). Les fonctions de cette interface de type `Shareable` sont les seules fonctions de l'API d'Air France utilisables par les autres applets, donc par le CBP. Ainsi, Air France peut partager les données concernant ses passagers avec le CBP au sein d'un mécanisme sécurisé et contrôlé par le JCRE (Java Card Runtime Environment).

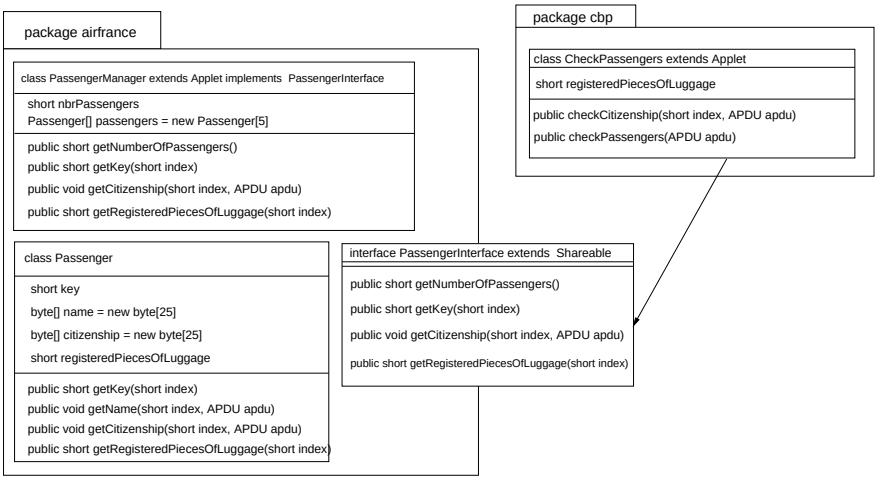

FIG. 4 – *Le partage des données.*

Une fois tous les passagers saisis par Air France, le CBP va tenter de déterminer quels sont ceux qui paraissent suspects. Pour cela, le CBP spécifie les critères nécessaires pour décider si tel passager est suspect ou non. Nous avons donc réalisé un outil graphique permettant une fois les critères saisis de les transmettre aux *applets* `CheckPassengers`, représentant le CBP sur les différentes cartes. Les *applets* `CheckPassengers` analysent alors la liste des passagers disponibles afin de trouver les passagers suspects correspondants aux critères de recherche. Chaque fois qu'un suspect est trouvé, la carte récupère son identifiant, `key` et l'envoie à une *applet* spéciale, appelée `CBPServer` qui centralise les résultats. Enfin, quand toutes les cartes ont fini la recherche, le CBP récupère auprès de l'*applet* `CBPServer` la liste des clefs des suspects trouvés et revient vers Air France pour demander les noms et prénoms de ces passagers, qu'Air France est libre de communiquer ou pas.

Pour être totalement exhaustif, on notera que dans notre application le CBP doit faire un minimum confiance à Air France, en ce sens qu'Air France aurait pu instrumenter son API pour tenter de découvrir les critères de recherche employés. De son coté s'il n'a pas totalement confiance le CBP peut aussi faire des requêtes inutiles pour tenter de leurrer Air France.

5 Travaux en cours

Nous travaillons actuellement sur le développement de nouvelles applications, le déploiement automatique des codes, l'augmentation significative de la taille de la plate-forme pour accroître la capacité de calcul et la gestion de *swap* sécurisé *off card* pour accroître la taille des données manipulables. Au delà du passage à un grand nombre de cartes, nous prévoyons l'utilisation des processeurs puissants issus des recherches actuelles et ayant une architecture de sécurité comparable (Trusted Computing Group), afin de les intégrer dès qu'ils arriveront sur le marché.

6 Conclusion

Dans cet article nous avons présenté nos travaux portant sur la sécurité et la confidentialité dans le cadre de l'extraction des connaissances. Nous avons montré les problématiques soulevées en ce qui concerne les données et les méthodes d'extraction.

L'application permettant le partage d'information entre les compagnies aériennes de l'Union Européenne et le CBP américain, que nous avons développée et décrite dans cet article, met en évidence l'apport à cette problématique d'une architecture distribuée à base de Java Cards.

Grâce à cette expérience, nous pensons pouvoir rapidement passer à l'échelle en terme d'une part de la quantité de données traitées et d'autre part de la complexité des traitements effectués, dès que des processeurs puissants adaptés seront présents sur le marché. Il n'en reste pas moins que l'architecture à base de cartes telle que nous l'avons mise en œuvre aujourd'hui offre une solution à des problèmes effectifs, comme nous l'avons vu dans cet article.

Remerciement

Notre projet est soutenu par Axalto, Gemplus et IBM BlueZ Secure Systems (pour les cartes), par SCM Microsystems et SmartMount (pour les lecteurs), et par Sun microsystems (pour la totalité de la plate-forme).

Nous remercions également : Fujitsu, Giesecke&Devrient, Oberthur Card Systems et Sharp pour les échantillons de Java Card ; David Corcoran et Ludovic Rousseau pour leur travaux sur `pcsc-lite` et le pilote générique CCID.

Références

Atallah, E., S. Chaumette, F. Darrigade, A. Karray, et D. Sauveron (2005). A Grid of Java Cards to Deal with Security Demanding Application Domains. In *Proceedings of e-Smart 2005*, Nice, France.

Chaumette, S., P. Grange, , A. Karray, D. Sauveron, et P. Vignéras (2005). Secure distributed computing on a Java Card grid. In *Proceedings of 7th International Workshop on Java for Parallel and Distributed Computing*, Denver, CO, USA.

Chaumette, S., P. Grange, D. Sauveron, et P. Vignéras (2003). Computing with Java Cards. In *Proceedings of CCCT'03 and 9th ISAS'03*, Orlando, FL, USA.

Chaumette, S. et D. Sauveron (2003). The Smart Cards Grid Project. `http://www.labri.fr/Perso/~chaumett/recherche/cartesapuce/smartcardsgrid/documents/poster.pdf`. Poster presented at Cartes 2003.

Chaumette, S. et P. Vignéras (2003). A framework for seamlessly making object oriented applications distributed. In *Parallel Computing 2003*, Dresden, Germany.

Chen, Z. (2000). *Java Card™ Technology for Smart Cards : Architecture and Programmer's Guide*. The Java™ Series. Addison-Wesley.

Corcoran, D., L. Rousseau, et D. Sauveron. pcsc-lite home page. `http://alioth.debian.org/projects/pcsclite/`.

Cour de Justice des Communautés Européennes (2005). Conclusions de l'avocat général dans les affaires c-317/04 et c-318/04. http://www.curia.eu.int/fr/actu/communiques/cp05/aff/cp050098fr.pdf.

Govindavajhala, S. et A. Appel (2003). Using Memory Errors to Attack a Virtual Machine. In *Proceedings of IEEE Symposium on Security and Privacy*.

IBM BlueZ Secure Systems. Java Wrappers for PC/SC. http://www.musclecard.com/middleware/files/jpcsc-0.8.0-src.zip.

Karray, A. (2004). Calcul sécurisé sur grille de cartes à puce. Master's thesis, ENIS – University of Sfax.

Parlement Européen (2004). Proposition de résolution de transfert des pnr. www.europarl.eu.int/meetdocs/committees/libe/20040309/524914fr.pdf.

PC/SC Workgroup. PC/SC Workgroup Home. http://www.pcscworkgroup.com/.

Rousseau, L. CCID free software driver. http://pcsclite.alioth.debian.org/ccid.html.

Sun microsystems (2003). *Java Card™ 2.2.1 Specifications*. Sun microsystems.

Transportation Security Administration. http://www.tsa.gov/public/display?theme=5&content=09000519800cf3a7, http://www.tsa.gov/public/interweb/assetlibrary/Secure_Flight_PRA_Notice_9.21.04.pdf.

Trusted Computing Group. Trusted Computing Group Home. https://www.trustedcomputinggroup.org/home.

Vignéras, P. et P. Grange. The Mandala website. http://mandala.sourceforge.net/.

Summary

Within the framework of data mining where the data and the code belong to different owners who do not necessarily trust each other, it seems difficult to reconcile the confidentiality and the integrity of the data on the one hand and those of the code and its execution on the other hand. It is a solution with this noncommonplace problem which we bring thanks to the use of data warehouses distributed on the grid of smart cards that we have developed. In this document we describe the exploitation of our work within the framework of a particular application.