Un système autonomique basé sur des bases de connaissances pour améliorer les performances d'un entrepôt de données

Vlad Nicolicin-Georgescu[1, 2], Vincent Benatier[2], Remi Lehn[1] et Henri Briand[1]

[1] LINA CNRS 6241 - COD Team - Polytech'Nantes,
Site Ecole Polytechnique de l'université de Nantes, Rue Christian Pauc , 44306 Nantes, France,
henri.briand@univ-nantes.fr
[2] SP2 Solutions,
8 Rue Rene Coty, 85000 La Roche sur Yon, France,
www.sp2.fr, vladgeorgescun@sp2.fr

Résumé : L'augmentation de la quantité et de la complexité des informations pose de plus en plus de problèmes pour les performances des entrepôts de données, en particulier ceux utilisés pour la prise de décision. Les experts en informatique décisionnelle rencontrant des entrepôts de plus en plus complexes, il devient nécessaire de disposer de dispositifs de gestion autonomiques. L'association de gestionnaires autonomiques avec des bases de connaissances afin d'en assurer la gestion est une solution de plus en plus utilisée en informatique décisionnelle. Dans ce cadre, nous proposons un système autonomique permettant d'analyser et d'optimiser les allocations de caches mémoire des entrepôts de données dans des environnements clients. Le système permet de représenter toutes les connaissances mises en œuvre dans la prise de décision dans le même entrepôt de données sous la forme d'ontologies. Il analyse les performances en cours à partir des statistiques d'exploitation du cache et propose des nouveaux paramètres d'allocation afin d'améliorer les performances.

Mots-clés : informatique décisionnelle, systèmes d'aide à la décision, calcul autonomique, entrepôts de données, ontologies, cache, règles de gestion.

Contexte: Communication appliquée

1 Introduction

Les connaissances constituent l'un des enjeux clés du 21ème siècle (A. Toffler, 1991). L'évolution des technologies de l'information rendent possibles la recherche, le traitement et la découverte automatique de connaissances à partir de l'ensemble des informations disponibles. Notre objectif est de développer un système permettant aux entreprises d'analyser et d'optimiser leurs processus de prise de décision. Les systèmes d'aide à la décision (*SID*) correspondent à des systèmes informatisés dont l'objectif est d'analyser des ensembles de faits et de proposer des actions concernant ces faits (M.J. Druzdel & R.R. Flynn, 1999). Les processus de prise de décision exploitant de tels systèmes et les éléments sur lesquels ils sont basés font partie du domaine de l'informatique décisionnelle.

Dans le cadre des SID, cet article met l'accent sur l'allocation de caches mémoire pour des entrepôts des données (bases multidimensionnelles ou cubes) *Oracle Hyperion Essbase BI*[1]. Ceci correspond à un des problèmes courants que rencontrent les experts en informatique décisionnelle. L'objectif est d'améliorer les performances de ces cubes avec une bonne configuration des caches. Le système proposé est basé sur des connaissances décrivant : le système et son architecture, les valeurs des caches et des jeux de règles représentant des contraintes et des conseils pour l'allocation des caches. Ces règles sont traduites des documents de maintenance d'*EssBase* et des connaissances de nos experts humains.

Deux aspects sont à prendre en compte particulièrement dans cette approche. Tout d'abord, les connaissances qui sont représentées par des données concernent des architectures matérielles et logicielles, les mesures de performances des systèmes et traduisent des pratiques d'analyse et d'optimisation. Ces données sont parfois décrites comme des règles "évènement condition action" (M.C. Huebscher & J.A. McCann, 2008). Ces éléments sont formalisés comme des bases de connaissances. Ensuite, le processus de prise de décision, nécessitant une amélioration afin de correspondre aux besoins du client. Pour parvenir à ce type d'objectifs, IBM a proposé une solution permettant d'aider à l'automatisation d'activités de gestion des systèmes: le calcul autonomique (IBM Co., 2005; M.C. Huebscher & J.A. McCann, 2008).

La section 2 détaille la base de connaissances que nous utilisons dans notre système pour piloter l'entrepôt de données. La section 3 développe la manière dont nous appliquons le calcul autonomique aux entrepôts de données pour permettre leur analyse et optimisation. La section 4 décrit la représentation physique de l'entrepôt de données et expose les résultats que nous avons obtenus. Enfin, en conclusion, nous montrons comment nos propositions permettent d'optimiser les processus en informatique décisionnelle et nous présentons nos directions de recherche.

2 Bases de connaissances – gestion et proposition

La gestion de connaissances est un processus permettant de formaliser, dans une représentation unifiée, partageable et exploitable, des informations à même de permettre un raisonnement expert à partir de données collectées auprès des centres opérationnels d'une entreprise (J. Oliveira & al., 2006). Ainsi, un entrepôt de données représente un référentiel des données stockées par une organisation et est conçu pour simplifier l'interrogation et l'analyse (WH Inmon, 2005). Nous proposons trois principaux types de connaissances impliqués dans la prise de décision.

L'information sur les systèmes et leur architecture correspond à toutes les données concernant n'importe quel matériel ou logiciel impliqué dans l'aide à la décision, par exemple la quantité de mémoire RAM disponible. La Fig 1 présente notre modèle hiérarchique de l'architecture du système décisionnel extraite de notre ontologie au moyen d'un diagramme de classes UML.

[1]http://download.oracle.com/docs/cd/E10530_01/doc/epm.931/html_esb_dbag/frameset.htm?dstcache.htm

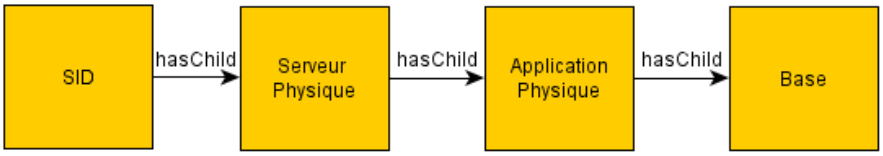

Fig. 1 – Architecture hiérarchique SID

Les rapports des performances des systèmes décrivent l'état du système. Ils contiennent de nombreux indicateurs de performance à prendre en compte tels que les temps de réponse sur les requêtes, le taux d'utilisation des caches, etc. Nous nous limitons aux indicateurs concernant l'utilisation des caches mémoire: le cache d'index, le cache sur les fichiers de données et le cache de données ainsi que le taux d'utilisation du cache comme mesure de performance. Le taux d'utilisation du cache représente le pourcentage des accès à des informations se trouvant dans le cache, relativement à l'ensemble des accès aux données. Bien que le problème de l'optimisation des performances au moyen de caches fasse débat (T. Malik & al, 2008; A.N. Saharia & Y.M. Babad, 2000), les solutions classiques sont soit au niveau de la conception du schéma de données, soit dans la mise en œuvre d'algorithmes permettant de déterminer quelles données sont les plus fréquemment utilisées. L'allocation de caches fait partie des principaux réglages des entrepôts de données, mais est peu abordée dans cette perspective.

L'analyse des systèmes et de leur fonctionnement correspond à un ensemble complexe de règles décrivant des interdépendances non triviales entre les éléments du système. Ses objectifs principaux sont de décrire les règles des processus d'analyse et d'optimisation. La représentation sous forme de règles permet une approche totalement différente de la gestion des connaissances, l'idée n'étant pas nouvelle (N. Stojanovic & S. Handschuh, 2002). En pratique, nous créons une base de connaissances de règles de gestion, exploitée dans les processus d'analyse et d'optimisation. Nous avons séparé les règles en deux parties. *Les conseils* déterminent les mesures du niveau d'amélioration du système (sous la forme d'une mesure numérique). Cette valeur dépend de la proximité de la configuration avec un conseil. *Les contraintes* représentent des limitations imposées, et des violations de ces contraintes entrainent des erreurs lors de l'analyse du fonctionnement des systèmes.

3 Calcul autonomique et la gestion de la base de connaissances

La plupart des organisations en informatique décisionnelle passent l'essentiel de leur temps au niveau de l'infrastructure, ce qui limite leur activité de supervision des systèmes, les empêchant de prévoir les problèmes avant que ceux-ci n'aient un impact auprès des utilisateurs (E. Manoel & al., 2005). Le sujet a déjà été appliqué à l'optimisation des performances des bases de données par IBM (V. Markl & al., 2003; S.S.Lightstone & al., 2002) et Microsoft (A. Mateen & al., 2008).

Les spécifications d'IBM définissent un composant, le *gestionnaire autonomique*, dont le rôle est de coordonner l'ensemble de l'activité du processus de calcul autonomique, de la détection de changements à la proposition d'actions. L'activité du gestionnaire autonomique est découpée en quatre phases : supervision, analyse,

planification et exécution (MAPE-K boucle, IBM Co., 2005). Nous l'avons appliqué à nos bases des connaissances, principalement au niveau de l'analyse du système et de son fonctionnement. Étant donné qu'il est défini à partir de règles, nous définissons des ensembles de règles pour chacune de ces phases. D'autres approches, non basées sur le calcul autonomique ont été proposées dans le cadre de l'informatique décisionnelle (T.M. Nguyen, 2005) mais avec la même idée : une boucle passant par différents états qui analyse et optimise un système donné.

La boucle MAPE-K fermée est régulièrement évaluée lors de la modification de valeurs de caches pour chaque base. Elle est évaluée chaque nuit lors de traitements par lots, alors que les statistiques sur le taux d'utilisation du cache sont collectées le jour. À chaque itération, les paramètres de cache sont ajustés jusqu'à obtenir un taux d'utilisation du cache demandé. La Table 1 montre l'évolution du taux moyen d'utilisation du cache pour l'ensemble des bases du système décisionnel, pour une période de deux semaines, sans prendre en compte les métriques d'optimisation.

Table 1 - Évolution du taux d'utilisation du cache sans métrique d'optimisation

Période /boucle AC	1	2	3	4	5	…	13	14
Taux d'utilisation du cache	0.5	0.5	0.56	0.53	0.6	…	0.69	0.7

Le temps mis à déterminer de nouvelles valeurs de cache est important. Nous proposons donc des règles de gestion à base de métrique (voir section 2) dans la boucle, afin que les mises à jour des paramètres de cache soient plus efficaces et plus pertinentes. Ainsi, le temps pour atteindre une valeur de taux d'utilisation du cache correcte est fortement réduit.

4 Approche – conception et résultats

Nous présentons dans cette section notre approche et les résultats de notre modèle. Nous utilisons un jeu de test dans lequel nous simulons un environnement du monde réel avec ses paramètres associés. Pour les types des données et de liens que nous devons faire entre ces données nous avons choisi d'utiliser les ontologies (T. Gruber, 1992) comme moyen de représentation. Il existe des travaux dans cette direction (L. Stojanovic et al., 2004) et nous utilisons ce modèle à la fois comme représentation de connaissances et pour y appliquer des règles. Les modèles de connaissances sont abordés selon deux points de vue : statique et dynamique.

Le point de vue statique de la base de connaissances est représenté par les deux types d'information : l'information sur les systèmes et les rapports de performances des systèmes. En pratique, cette base de connaissances est décrite au moyen de classes, individus et propriétés les associant entre eux. Nous utilisons le langage OWL 2[2] comme langage d'ontologie. Les valeurs de cache et de taille de fichiers sont exprimées grâce à des « datatype properties » OWL, étant données que ce sont des valeurs scalaires (nombres décimaux). Les concepts sont mis en relation au moyen d'object properties OWL. Dans la Figure 1, nous pouvons voir par exemple la relation hasChild qui met en relation les entités pour décrire complètement le système

[2]http://www.w3.org/2007/OWL/wiki/OWL_Working_Group

décisionnel. Au total, l'ontologie regroupe actuellement plus de 150 axiomes, 16 classes, 35 individus, 15 propriétés objet et 46 propriétés data type.

Le point de vue dynamique constitue le principal enjeu de notre approche et fournit les principales innovations. Il regroupe toutes les règles correspondant à la boucle du calcul autonomique, les états du gestionnaire autonomique, les règles déterminant les dépendances entre les propriétés, et les inférences d'instances de l'ontologies. Pour des raisons de simplicité et d'efficacité, nous avons choisi d'utiliser des règles Jena via l'API Java Jena[3] pour le développement de l'ontologie. Les règles sont réparties selon les connaissances qu'elles produisent, conformément à la boucle du calcul autonomique. Au total, il y un nombre de 56 règles, qui regroupent les règles métiers et les règles de gestion des phases du gestionnaire autonomique.

Prenons la règle d'analyse métier décrite de manière informelle de la façon suivante: pour une base, la valeur du cache d'index doit être la plus proche possible de la taille de fichier d'index (dans l'idéal si nous stockons tous les index en cache, nous avons un taux d'utilisation de cache de 100%). Nous formalisons cette règle et nous obtenons des métriques pour les différentes proportions : *[rule: (?base rdf:type cp:c_Base) (?base cp:dp_hasIndexFileSize ?ifs) (?base cp:dp_hasIndexCache ?ic) quotient(?ic, ?ifs, ?rap) ge(?rap, "0.95"^^xsd:double) -> (?base cp:dp_hasPoints_Advice_IndexCacheAllocation "1000"^^xsd:int)]*

Nous avons effectué plusieurs tests et nous avons comparé l'amélioration du taux d'utilisation du cache selon que l'on utilise ou pas les règles métier. De plus, nous avons considéré que, pendant le processus d'optimisation, nous ajoutons une base de données (Table 2, jour 8). L'allocation de cache pour cette nouvelle base est la valeur par défaut préconisée par l'éditeur. Considérant la boucle normale, sans les règles métier, la reconfiguration des caches afin d'atteindre le niveau escompté prend beaucoup de temps. Lorsque nous utilisons les règles métier, ce temps est nettement moins important.

Table 2 – Évolution du taux d'utilisation du cache avec / sans règles métier

Période (nuit)/boucle AC	1	2	3	…	8	…	13	14
TdUC **sans** BR	0.5	0.5	0.56	…	0.5	…	0.62	0.63
TdUC **avec** BR	0.65	0.68	0.7		0.65		0.7	0.7

En pratique, à chaque demande de configuration système (nuits 1 et 8), nous récupérons la meilleure configuration de cache possible du point de vue de la métrique d'amélioration. De cette manière, nous démarrons le processus d'allocation de cache à partir d'une meilleure configuration. Apres 14 jours on voit bien que le processus normal sans les règles métier n'est pas capable de donner une configuration pour obtenir le taux de 0.7. En revanche, le système avec les règles métier est déjà arrivé à cette configuration depuis quelques jours.

5 Conclusions

Dans cet article, nous avons présenté comment le calcul autonomique et les ontologies permettent la gestion de systèmes décisionnels et, plus spécifiquement, l'ajustement de paramètres d'allocation de caches dans les cubes Hyperion Essbase.

[3]http://jena.sourceforge.net/inference/#rules

Ce n'est pas la première fois que les ontologies et le calcul autonomique sont utilisés conjointement (J.M. Gonzales & al., 2007), mais notre originalité est que nous les appliquons à l'informatique décisionnelle et en utilisant des règles de gestion. Notre objectif est d'intégrer le prototype présenté ici avec l'ensemble des paramètres d'un système décisionnel.

Nos perspectives concernent l'extension du domaine d'étude au-delà du dimensionnement des caches en utilisant une ontologie beaucoup plus enrichie. Le domaine étant relativement nouveau, peu de choses ont été écrites à ce sujet, nous tentons donc de contribuer le plus possible au développement de systèmes autonomiques pour l'aide à la décision.

Références

A. MATEEN, B. RAZA, T. HUSSAIN (2008), *Autonomic Computing in SQL Server*, 7th IEEE/ACIS International Conference on Computer and Information Science, 2, p. 113 – 118

A. N. SAHARIA, Y.M. BABAD (2000), *Enhancing Data Warehouse Performance through Query Caching*, The DATA BASE for Advances in Informatics Systems, Vol 31. No.3

A. TOFFLER (1991), *Powershift: Knowledge, Wealth and Power at the edge of the 21st Century*, Bantam Book Publishing

E. MANOEL, M.J. NIELSEN, A. SALAHSHOUR, S. SAMPATH, S. SUDARSHANAN (2005), *Problem determination using self-managing autonomic technology*, IBM RedBook, p. 5 - 9

IBM CORPORATION (2005), *An architectural blueprint for autonomic computing*, p. 9-18, et *Autonomic Computing. Powering your business for success*, International Journal of Computer Science and Network Security, VOL.7 No.10, p. 2-4

J. OLIVEIRA, J.M. DE SOUZA, M. PERAZOL (2006), *Managing knowledge about resources for autonomic computing*, 1st latin american autonomic computing symposium, p. 124-126

J.M. GONZALES, J.A. LOZANO, J.E. LOPEZ DE VERGARA, V. A. VILLAGRA (2007), *Self-adapted service offering for residential environments*, ACNM'07, p. 48-55

L. STOJANOVIC, J. SCHNEIDER, A. MAEDCHE, S; LIBISCHER, R. STUDER, TH. LUMPP, A. ABECKER, G. BREITER, J. DINGER (2004), *The role of ontologies in autonomic computing systems*, IBM Systems Journal, Vol. 43, No. 3, p. 598-616

M.C. HUEBSCHER, J.A. MCCAN, *A Survey on Autonomic Autonomic – Degrees, Models and Applications*, ACM Computing Surveys, Vol. 40, No. 3, Article 7, August 2008

M.J. DRUZDEL, R.R. FLYNN (1999), *Decision Support Systems*, Encyclopedia of library and information science

N. STOJANOVIC, S. HANDSCHUH (2002), *A framework for knowledge management on the semantic web*, The 11th International WWW Conference, 2002

S.S. LIGHTSTONE, G. LOHMAN, D. ZILIO (2002), *Toward autonomic computing with DB2 universal database*, ACM SIGMOD Record Volume 31, Issue 3

T. GRUBER (1992), *What is an ontology?* Academic Press Pub.

T. MALIK, X. WANG, R. BURNS, D. DASH, A. AILAMAKI (2008), *Automated Physical Design in Database Caching*, ICDE Workshop 2008

T. M. NGUYEN, J. SCHIEFER, A. MIN TJOA (2005), *Sense & Response Service Architecture (SARESA)*, DOLAP'05

V. MARKL, G. M. LOHMAN, V. RAMAN (2003), *LEO : An Autonomic Optimizer for DB2*, *IBM Systems Journal*, Vol. 42, No. 1, 2003