An Ontology-Based Autonomic System for Improving Data Warehouse Performances

Vlad Nicolicin-Georgescu, Vincent Benatier, Rémi Lehn, Henri Briand

An Ontology-Based Autonomic System for Improving Data Warehouse Performances

Vlad Nicolicin-Georgescu[1, 2], Vincent Benatier[2], Remi Lehn[1] and 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

Abstract: With the increase in the amount and complexity of information, data warehouse performance has become a constant issue, especially for decision support systems. As decisional experts are faced with the management of more complex data warehouses, a need for autonomic management capabilities is shown to help them in their work. Implementing autonomic managers over knowledge bases to manage them is a solution that we find more and more used in business intelligence environments. What we propose, as decisional system experts, is an autonomic system for analyzing and improving data warehouse cache memory allocations in a client environment. The system formalizes aspects of the knowledge involved in the process of decision making (from system hardware specifications to practices describing cache allocation) into the same knowledge base in the form of ontologies, analyzes the current performance level (such as query average response time values) and proposes new cache allocation values so that better performance is obtained.

Keywords: Business Intelligence, Decision Support Systems, Autonomic Computing, Data Warehouse, Ontology, Cache, Business Rule

1 Introduction

As the 21st century is well on its way, in a civilized and modern world, we realize that the most important asset needed in order to keep the pace with this new rhythm is knowledge. Knowledge is the source of power and truly the new edge of the power shift [4]. As technology comes greatly to our help, we find it normal to research, discover and improve ways of gathering, processing and using all information available. Our purpose is to develop a system aimed at helping enterprises analyze and improve their decision making process by providing a unified representation of certain aspects involving the knowledge available for this process (from software support documents to human expert experience) and an autonomic system that makes use of this knowledge and acts upon it. Simpler and often referred to as DSSs, *Decision Support Systems* are defined as computerized systems whose main goal is to

analyze a series of facts and give various propositions for actions regarding the facts involved [12]. This is why the process of decision making, based on such systems and the elements involved, is known as *business intelligence* (BI) process.

The applicative area of this paper is *cache memory* allocation for the Oracle Hyperion Essbase BI[1] cubes. This is a common configuration problem that BI experts are faced with. The cubes represent the data warehouse whose performances are to be improved. The system we propose makes use of knowledge based on system information (from architecture to cube cache parameters) and on sets of rules representing constraints and advice for the cache allocations (taken from the Essbase documents and from our human experts). The purpose is to provide two main functionalities. First, to compute a system's degree of improvement based on cache allocations and performance indicators. Second, to propose an improved cache configuration, that gives (if possible) the optimal performances. Two main aspects have been taken into consideration along with this approach.

The first aspect is knowledge representation. The knowledge regroups several sources: describing software and hardware architectures, system performance measurement, system analysis and improvement practices (described as sets of ECA (event condition action) rules [11]). Knowledge representation describes how this information is unified into knowledge bases. If for the system architecture, models are being developed and even adopted as w3c standards[2], then for the data representation of system report performances and the rules of system analysis, we are obliged to turn to specific representations (using ontologies [16] and ontology based rules).

Second, the improvement process itself, meaning having a fast response (from the moment a demand for improvement is made) and having a good (if not the best) response for any type of decision request (in our case a new cache allocation). In order to achieve this, IBM has proposed a solution to help automate various processes. The solution is called *Autonomic Computing* [6], [11] and its applications extend way beyond the business intelligence sector. We propose the usage of autonomic computing with the cache allocation improvement process.

Section 2 gives an insight of how we manage the knowledge in our system in order to drive the data warehouse. First we present how the knowledge base is organized for managing data warehouses and then how autonomic computing is used in the decision making process. Section 3 focuses on the description of our model and how this approach is used to perform analysis and improvement. A schema of a DSS together with the description of the data warehouses and an example of associated rules are presented. Section 4 provides a view of the experimentation and the results obtained. Finally, we sum up the work presented and take a glance at the future directions.

---

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

[2] http://www.w3.org/TR/2008/CR-sml-20081125/

2 Data Warehouse Management

2.1 Knowledge Management

In brief, Knowledge Management is the process through which organizations generate value from their intellectual and knowledge-based assets, disseminating this knowledge and sharing it in an effort to get competitive advantage [7]. Data warehouse (DW) management is a key element in the decision making process. A data warehouse is a repository of an organization's electronically stored data and is designed to facilitate reporting and analysis [20]. Managing a data warehouse includes the process of analyzing, extracting, transforming and loading data and metadata. Our interest in knowledge management comes from the types of data involved in the decision making process.

The knowledge management into our work is based on system analysis and functioning. These refer to a complex set of rules that describes the functioning and non trivial interdependencies between the elements of the system. The main objective is to describe the rules for the analysis and improvement processes. Representing data under the form of rules [14] gives a completely different approach to knowledge management. Practically, we create a business rule knowledge base that serves for the process of analysis and improvement. This process is supported both by the human expert and by the autonomic system. We propose to divide these rules into two main components: constraints and advice.

Advice represents business rules (BR) and best practices for the DSS giving the measure of a system improvement level in these terms. This means how 'close' a configuration is to satisfy sets of advice and therefore is able to generate an advice scoring. This scoring is built upon a point allocation system that grades the level of implementation of an advice set which we call a BR improvement points system, and which we describe in Section 3.

Constraints represent limitations imposed (i.e. the index cache cannot be under 1 Mb) and a violation of such constraints leads to an error in the system analysis.

To the division above, an entire set of rules is added, (from initial fact deduction to planning and action rules). These sets of rules are considered as state specific rules and are modeled for each state of the autonomic computing manager (presented in the next section).

2.2 Autonomic Computing in Data Warehouse Management

Most of the IT organizations spend a lot of time reacting to problems that occur at the IT infrastructure component level. This prevents them from focusing on monitoring their systems and from being able to predict and prevent problems before end users are impacted [5]. Autonomic computing (AC) is the ability for an IT infrastructure to adapt and change in accordance with business policies and objectives. Quite simply, it is about freeing IT professionals to focus on higher–value tasks by making technology work smarter, with business rules guiding systems to be self-configuring, self-healing, self-optimizing and self-protecting [6]. This subject is of great interest to enterprises and has already been put into practice for improving database performance by IBM

[19], [15] and Microsoft [2]. There is great interest of development into applications of autonomic computing on managing data warehouses, as experts can no longer face the quantity of information available

IBM specifications link autonomic computing with the notion of *autonomic manager* as the entity that coordinates the activity of the autonomic process. Four separate phases are distinguished for the manager: monitoring, analyzing, planning and executing [6], [10]. We propose an implementation of the autonomic manager connected to our knowledge base and based on the data warehouse performance. As it is rule based knowledge, we differentiate the sets of rules for each of these phases. The illustration of this process is shown in Section 3. Similar alternatives to autonomic computing were made in real BI [18] but the idea is the same: to be able to analyze and improve (in our case) a given system through a closed loop that differentiates a series of states.

The loop formed by the four states mentioned is regularly run. By regularly we mean once per night during batch operations, when statistics on the data warehouse usage for that day are gathered. Each loop, according to the new query times, modifies the values of the caches, and this is repeated until the desired times are achieved. This process is based on both the feedback from the previous response times and on the sets of advice mentioned earlier. Consequently, what our solution proposes is to include business rule in the loop so that the modifications to the cache values are more substantial and relevant, and thus the time needed to reach the desired performance level is greatly reduced.

3 Knowledge Base

We have seen the information we need to formalize and we have found the ontology [16] as a model of knowledge representation. The ontology representation suits our needs as it provides the solution for two main problems: knowledge unification and knowledge interchange. Works in this area have already been done by [9] and we found this model fully applicable to our system as it covers both knowledge formalization and rule usage. We propose a division of the knowledge aspect into two main categories: static and dynamic.

3.1 Static knowledge base

The static aspect of knowledge contains all the knowledge representation under the form of ontology concepts: classes, individuals and the properties linking them. Our implementation uses OWL [3] as ontology description language and Protégé 4[4] as software support for ontology manipulation. The ontology contains over 150 concepts and over 250 axioms. We propose two main data types for the static knowledge base:

System information and architecture refers to all data concerning the software and hardware specifications of the DSS system (i.e. the quantity of RAM memory

[3] http://www.w3.org/2007/OWL/wiki/OWL_Working_Group
[4] http://protege.stanford.edu/download/registered.html

installed on a server). This subject has been approached [1] and detailed by a certain number of editors [13]. **Fig. 1** shows how we model the DSS architecture hierarchically, with the use of a UML[5] diagram. Several entities are distinguished, starting from the top of the hierarchical tree (the DSS) and, at each level one or more sub-components can be identified with the specific parameters.

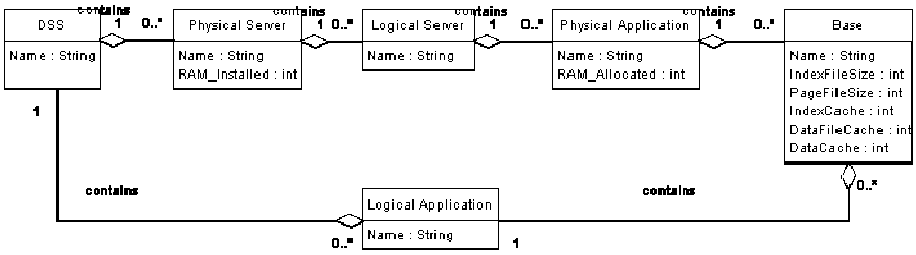

Fig. 1 DSS hierarchical architecture

System report and performance contains aspects regarding the DSS, in particular the data warehouse, parameters and performance indicators. There are many indicators to take into consideration: memory cache values, query response times, report editing times, aggregation operation times, etc. This paper considers three types of the Essbase cube cache: Index Cache, Data File Cache and Data Cache and the query response time as performance indicator. The implication is that by a good configuration of these three caches according to a given situation (i.e. at night some bases are not used, some cache allocations are useless, etc.) we gain substantial processing time. Although the problematic of performance improvement in data warehouses throughout caches is debated [17], [3] the issue is always addressed either through the physical design or the design of algorithms to determine which information is likely to be stored in cache memories. Cache allocation improvement is an important aspect of data warehouse tuning and there are little or no works on cache improvement in the context of data warehouses.

3.2 Dynamic knowledge base

The dynamic aspect is the challenge in our work and provides the main innovating approaches. It formalizes the functioning system and the analysis aspect of data warehouse management presented in 2.1. It contains all the rules that are part of the AC loop and all the rules that determine property interdependence and individual inference in the ontology. For reasons of simplicity and efficiency we have chosen to use Jena Rules via the Jena Java API[6] for ontology development. The rules are divided according to their area of activity in conformity with the AC loop phases.

For the business rule illustration, we give below an example of an analysis business rule that informally states: *The closer the Index Cache value is to the Index*

[5] http://www.uml.org/

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

File Size for a base, the better. We formalize this rule and obtain a discrete point allocation for the different proportions:

*[**rule1**: (?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)]
*[**rule2**: (?base rdf:type cp:c_Base) (?base cp:dp_hasIndexFileSize ?ifs) (?base cp:dp_hasIndexCache ?ic) quotient(?ic, ?ifs, ?rap) lessThan(?rap, "0.95"^^xsd:double) ge(?rap, "0.85"^^xsd:double)*
-> (?base cp:dp_hasPoints_Advice_IndexCacheAllocation "900"^^xsd:int)] ...

The formalization process of business rules such as this one a very important aspect, and always requires an expert hand. By applying all the analysis rules we obtain an overall scoring and we can calculate a performance level of all the data warehouses from the point of view of business rules.

For the autonomic computing rule illustration we propose a simple example of the passage through the 4 states of the autonomic manager. We show the index cache is modified in a cycle.

Monitor: retrieval of the response time and the current cache values for a DW. These are stored in the knowledge base via the java program.

Analyse: we compare the average response time of the DW with its desired response time. If it is greater, the caches must be increased:

(?base cp:dp_hasAvgResponseTime ?avgt) (?base cp:dp_hasDesiredResponseTime ?dt) ge(?avgt, ?dt) -> (?base cp:hasState cp:IncreaseCache)

Plan: we try to see if the increased cache state can be applied for the index cache. If the new cache value (increased with 10% of its current value) is not greater than the allocated memory, then plan the change to the index cache:

(?base cp:hasState cp:IncreaseCache) (?base cp:dp_hasIndexCache ?ic) (?base cp:dp_hasAllocatedMemory ?am) product(?ic, '1.1'^^xsd;double, ?newic) le(?newic, ?am) -> (?base cpdp_:dp_hasIndexCache ?newic)

Execute: Execute a modification script with the new proposed value of the index cache. This is done via the java program.

4 Experimentation and Results

For our experiments we have considered a test suite that simulates a real environment with the associated parameters. On an existing server we have chosen an Essbase cube as the data warehouse whose performances were improved. For the pertinence of the tests, the cube was created starting from the "Sample" base provided by Essbase. The cube contains in average 11 principal axes and 27 level 2 axes and the data file has an average size of 300MB.

With the respective cube we carried out several tests corresponding to several configurations. We had to simulate the night/day loop passage faster so we made a series of 5 queries (from very fast to very slow as time of response) and applied them to each configurations. This process was iterated 10 times therefore simulating a day/night cycle. At the end of each cycle we fetch the average response times for each of the bases and pass through the autonomic computing loop so that we could optimize the cache allocation where it was necessary. The evolution of cache

allocation with the response times can be observed in **Fig. 2**. We can see the difference between the minimum cache allocations in configuration C1 which is almost 6 times slower than the maximum possible allocation in configuration C6. As a maximum allocation is not always possible due to the quantity of memory available, we have to try our best to improve the performances. Applying our approach, configuration C3 maximizes the BR improvement points, and with a passage through the AC loop several times we get a configuration which has a smaller response time C4. The main improvement is that C4 is using only 174MB for its caches whereas C3 requires 240MB. In addition, the average response times in comparison with a normal or random configuration (as in C2 and C3) are improved at least 2 times.

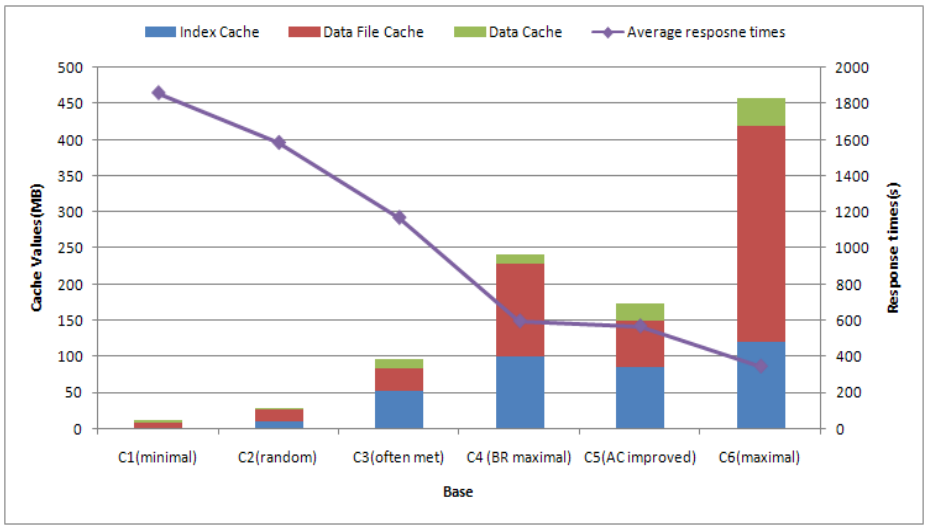

Fig. 2. Query average response times evolution with the cache configurations

5 Conclusions

This article presented how autonomic computing and ontologies can be used for helping DSS experts improve the cache memory allocations for data warehouses. It is not the first approach that tries to combine the two elements together [9],[8] but the premiere is its application in the field of business intelligence and data warehouse improvement using business rules.

There are many positive aspects to this approach of decision support system management: the simple and intuitive yet powerful ontology representation, the facilities of web semantics and rule support brought to the domain of BI and, last but not least, the process of autonomic computing to manage all these elements.

Our future directions are to expand the Data Warehouses described above so that our small prototype can prove its efficiency on more than a ‘few simple rules’. Our purpose is to integrate the prototype presented here with more than one aspect (data warehouse cache allocations based on response times) of decision support systems.

As the domain is relatively new and not much has been written on the subject yet, we try to bring as much support as possible for future development in the direction of autonomic decision support systems. We follow these changes and hope that our work will equally add something to this expanding environment.

References

1. A. Ackerman, J. Tyree, Using ontologies to support development of software architectures, IBM Systems Journal, Vol. 45, No. 4 (2006)
2. A. Mateen, B. Raza, T. Hussain, Autonomic Computing in SQL Server, 7th IEEE/ACIS International Conference on Computer and Information Science, p. 113 – 118 (2008)
3. A. N. Saharia, Y.M. Babad, Enhancing Data Warehouse Performance through Query Caching, The DATA BASE Advances in Informatics Systems, Vol 31, No.3 (2000)
4. A. Toffler, Powershift: Knowledge, Wealth and Power at the edge of the 21st Century, Bantam Book Publishing, (1991)
5. E. Manoel, M.J. Nielsen, A. Salahshour, S. Sampath, S. Sudarshanan, Problem determination using self-managing autonomic technology, IBM RedBook, p. 5 - 9 (2005)
6. IBM Corporation, 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 (2005)
7. J. Oliveira, J.M. de Souza, M. Perazol, Managing knowledge about resources for autonomic computing, 1st latin american autonomic computing symposium, p. 124-126, (2006)
8. J.M. Gonzales, J.A. Lozano, J.E. Lopez de Vergara, V. A. Villagra, Self-adapted service offering for residential environments, ACNM'07, p. 48-55 (2007)
9. L. Stojanovic, J. Schneider, A. Maedche, S; Libischer, R. Studer, Th. Lumpp, A. Abecker, G. Breiter, J. Dinger, The role of ontologies in autonomic computing systems, IBM Systems Journal, Vol. 43, No. 3, p. 598-616 (2004)
10. M. Parshar, S. Hariri, Autonomic Computing: Concepts, Infrastructure and Applications, Taylor and Francis Group (2007)
11. M.C. Huebscher, J.A. McCann, A Survey on Autonomic Computing – Degrees, Models and Applications, ACM Computing Surveys, Vol. 40, No. 3, Article 7, (2008)
12. M.J. Druzdel, R.R. Flynn, Decision Support Systems, Encyclopedia of library and information science, (1999)
13. Microsoft Corporation, Understanding system definition model (SDM) and its practical application in 2006 to 2008, p. 3-5, (2006)
14. N. Stojanovic, S. Handschuh, A framework for knowledge management on the semantic web, The 11th International WWW Conference, (2002)
15. S.S. Lightstone, G. Lohman, D. Zilio , Toward autonomic computing with DB2 universal database, ACM SIGMOD Record Volume 31, Issue 3, (2002)
16. T. Gruber, What is an ontology?, Academic Press Pub. (1992)
17. T. Malik, X. Wang, R. Burns, D. Dash, A. Ailamaki, Automated Physical Design in Database Caching, ICDE Workshop (2008)
18. T. M. Nguyen, J. Schiefer, A. Min Tjoa, Sense & Response Service Architecture (SARESA), DOLAP'05, (2005)
19. V. Markl, G. M. Lohman, V. Raman, LEO : An Autonomic Optimizer for DB2, IBM Systems Journal, Vol. 42, No. 1, (2003)
20. W.H. Inmon (1995), Tech topic: what is a data warehouse?, Prism solutions. Volume 1., (1995), Data warehouse performance, Wiley Publishing, p. 19-20, 209-304 (1999) and Building the data warehouse, fourth edition, Wiley Publishing, p. 29-33, 79-94, 331-33, (2005)