Optimized Data Management for E-Learning in the Clouds towards Cloodle

Mirna Adriani, Yeow Wei Choong, Ba-Hung Ngo, Laurent d'Orazio, Dominique Laurent, Nicolas Spyratos, Bruno Bachelet, Christophe Duhamel, Tao-Yuan Jen, Claudia Marinica, et al.

Optimized Data Management for E-Learning in the Clouds Towards Cloodle

Mirna Adriani
Faculty of Computer Science
University of Indonesia
Depok, Indonesia
mirna@cs.ui.ac.id

Yeow Wei Choong
HELP University
Kuala Lumpur, Malaisia
choongyw@help.edu.my

Ba Hung Ngo
Can Tho University
CNRS - UMR 8623 LRI
Cantho, Vietnam
nbhung@cit.ctu.edu.vn

Laurent d'Orazio
Clermont Université -
Université Blaise Pascal
CNRS - UMR 6158 LIMOS
Clermont-Ferrand, France
laurent.dorazio@univ-bpclermont.fr

Dominique Laurent
Université du Grand Ouest
Parisien - Université de Cergy Pontoise
CNRS - UMR 8051 ETIS
Cergy Pontoise, France
dominique.laurent@u-cergy.fr

Nicolas Spyratos
UniverSud Paris - Université Paris Sud
CNRS - UMR 8623 LRI
Orsay, France
nicolas.spyratos@lri.fr

ABSTRACT

Cloud computing provides access to "infinite" storage and computing resources, offering promising perspectives for many applications, particularly e-learning. However, this new paradigm requires rethinking of database management principles in order to allow deployment on scalable, easy to access infrastructures, applying a pay-as-you-go model in which failures are not exceptions but rather the norm. The GOD project aims to provide an optimized data management system for e-learning in the cloud by rethinking traditional database management techniques, extending them to consider the specificities of this paradigm.

Categories and Subject Descriptors

H.2.4 [**Systems**]: Distributed databases, Parallel databases; H.3.4 [**Systems and Software**]: Distributed systems

General Terms

Algorithms, Design

Keywords

Data Management, Cloud Computing, Big Data, Optimization, E-Learning

1. INTRODUCTION

Cloud computing [6] aims to tackle increasing needs of computing and storage resources, in a userfriendly way and has attracted increasing interest, in particular by major IT companies (Google, Microsoft or Amazon). Clouds thus enable to envison data management at an unexpected scale and in diverse contexts. In particular, clouds are quite promising for e-learning, referring to educational technology for learning and teaching.

However, clouds' performance usually relies on brute force (i.e. using more nodes and more powerful nodes) thus leading to high cost and suboptimal resource management. Performance optimization has been studied for years in databases using methods such as indexing, view materialization or cache. These methods could also help improve the performance in the cloud. In particular, materialized views and semantic cache could enable to rewrite queries so that to reuse results from previous requests. However, existing solutions, relying on well known languages (SQL, Xpath, etc.) or data models (relational, object, etc.), are not suited to cloud data management, which uses key-value stores or analysis systems based on large scale distributed file systems.

The GOD project aims to propose Cloodle, an optimized cloud version of the Moodle e-learning environment. The main contributions of Cloodle are an e-learning system relying on a cloud data back end, an adaptable semantic cache to be used with rewriting rules, cost- and preference-aware query processing.

The remainder of this paper is organized as follows. Section 2 motivates the GOD project. Then Sections 3 and 4 detail sophisticated caching strategies while Sections 5 and Section 6 focus on cost- and preference-aware query processing. Finally, in Section 7, we conclude the paper.

2. CONTEXT AND MOTIVATION

E-learning is a term used to include all forms of educational technology for both learning and teaching. This approach in learning and teaching is particularly suitable for distance learning since technologies can provide the solution when the environment is as such that the source of knowledge (i.e. information) and the learners are separated by

SoICT '13, December 05 - 06 2013, Danang, Viet Nam

http://dx.doi.org/10.1145/2542050.2542065

time or distance, or both.

The e-learning technologies have been evolving alongside with the advancement of the World Wide Web. As more and more courses are provided online through the Internet, one can study without attending classes. Learners are able to complete their studies whenever and wherever without having to travel or to participate at specific time e.g. classes according to the timetable provided. In the United States, over a third of all students enrolled in postsecondary education took an online course for credit in fall 2011 [7]. This new model requires a massive data management system.

One of the widely used e-learning platforms for institutions of higher learning is Moodle [1]. This platform is an Open Source Course Management System where developers are allowed to make modifications based on their needs with certain terms and conditions. Moodle provides blended learning opportunities as well as platforms for distance learning courses. Many universities and colleges use Moodle as the online learning system in their daily learning and teaching activities.

The main goal of the GOD project is to supply Cloodle, a cloud version of Moodle relying on an optimized data management system. The first task of this project consists in substituting the MySQL backend currently used in Moodle with a cloud data management system with respect to the data and the queries to be processed. Several systems exists. Some of them consists on large scale solutions providing a simplified query interface (usually a subset of SQL), like Amazon SimpleDB [3] or DynamoDB [1]. Less scalable but fully relational approach are also available, such as Amazon RDS [2] and SQL Azure [26]. Data intensive analysis tools such as Pig [28], Hive [32], SCOPE [10] or Jaql [8], relying on massively parallel execution environment like MapReduce [16], or its open source version Hadoop [5] have been developped.

Cloud data management system's performance usually relies on brute force, using more nodes or/and more powerful nodes. This leads to high cost and suboptimal resource management. Performance optimization has been studied for years in databases. In this project, we particularly aims at studying four techniques: semantic caching 3, rewriting 4, cost-based optimization 5 and querying with preferences 6.

3. SEMANTIC CACHING

Semantic caching [23, 14] manages a cache as a set of query results. When a query is posed, it is decomposed into two disjoint queries: the probe query retrieves the result already present in the cache and the remainder query retrieve missing objects.

Semantic caching has been studied in several contexts: distributed databases [23, 14], web [11, 12, 25] and grid computing [18]. These solutions differ in several ways. The cache can thus store query results [14] called *semantic regions* or *semantic segments*, objects to be strongly [23] or independently [18] associated to and possibly shared by predicates. Some of them focus on a specific data structure like XML [11], [25]. Efficient research, via *signature files* has been proposed for keyword based and conjunctive queries. In a previous work, we deployed P2P semantic caches, called CoopSC [35] to highlight potential money savings in the cloud, in addition to traditional time and bandwidth consumption reductions. All these techniques are complementary to our project and can be reused to provide finely tuned caches.

Using a cache framework, such as ACS [17], GOD data management system will rely on tunable semantic caches, particularly, two adaptable main functionalities: query analysis and query evaluation. Query analysis refers to the semantic process of comparing submitted query with the content of the cache to deduce semantic overlap or mismatch, and identifies one or more cache entries to be used to answer a submitted query. Query evaluation consists of operators (selection, projection, order, group by, etc.) to locally evaluate queries on the cache content.

4. REWRITING

The problem of optimizing semantically related queries has received a lot of attention during the last decades. Main references are presented into a wide survey on this topic [20]. Our approach differs from these solutions in two main ways. First, to the best of our knowledge, because [24] is the first approach to OLAP query rewriting in which the content of the cache is optimized. Second, our approach is meant to take into account specificities of NoSQL systems (considering not only the well known relational model) and their associated query languages.

Another approach to optimizing query processing is to pre-compute queries, also known as *materialized views*, store the corresponding answers in a cache and reuse them in the query evaluation process. In [24], we have proposed novel solutions to answer queries using views in a common formalism based on the partition semantics [29]. Every OLAP query Q is associated to a partition $\Pi(Q)$, whose blocks are the sets of tuples that are grouped together by the `GROUP BY` clause in Q when processing aggregates. Given a query Q, the set of all queries Q' associated with the same partition as Q can be characterized using the functional dependencies that hold on the database. In addition, partitions allow to compare queries and such comparisons can be expressed using functional dependencies. Contrary to all other approaches, using such theoretical result in our solution, the answer to Q is *not* stored , but it is augmented by (1) attributes occurring in the closure of the attributed set of the answer to Q and (2) attributes storing aggregate values not occurring in Q. First, this makes it possible to rewrite as many queries as possible using a given query Q with limited extra storage. Then a provided actual rewriting method enable to easily answer these queries. Finally, these rewriting techniques can be used to efficiently manage the cache content, storing mutually non redundant queries.

However, none of them takes into account properties such as elasticity and pay-as-you-go.

5. COST MODELS AND OPTIMIZATION

GOD will consider deploying e-learning systems in public clouds and a pay-as-you-go model. This model leads to consider data management in general, and optimization particularly, from a novel point of view. Recent approaches [15, 22, 34] have addressed the challenge of selecting the optimizations to implement and the way to price them in a data shared environment. These solutions studied the optimization in a shared environment at a higher level than we do in the GOD project. Indeed, cloud providers such as Amazon, Google, and Microsoft, supply a pool of resources, like

[1]https://moodle.org/

hardware (CPU, storage, networks), development platforms or services. Each provider offers different pricing and services. Then this pricing model may be consider from different points of view.

One of our preliminary prior work provided cost models, inspired from AWS's offer, for materialized views in the cloud [27]. In our simplified model, the total cost C for cloud data management is:

$$C = C_c + C_s + C_t. \quad (1)$$

In this model, C_c, C_s and C_t are the sum respectively of computing costs, storage costs and data transfer costs, depending on several parameters such as the size of the data set, the pricing model, the type and number of nodes or the storage time.

In this project, we will first propose a uniform cost model for data management in the cloud, taking into account the pricing model of the main providers (Amazon, Microsoft and Google) and extending it with their specificities (licences, internal data transfers, etc.).

The GOD project will then consider cost-based optimization in two distinct cases. The first one is about general resources management, to provide the accurate level of resources (type and number of virtual machines) for the e-learning platform to supply a given level of quality of service. The second one will focus on query processing in presence of semantic caches and rewriting rules.

We will use multi-criteria optimization to enable users specifying objectives: (1) minimizing the response time given a fixed budget, (2) minimizing the cost given a deadline or (3) finding a trade-off between time and cost. Exact solving methods, like Mixed Integer Programming (MIP) techniques, and metaheuristics, like GRASP, will be considered to solve mono-objective problems. Metaheuristics like NSGA-II will be studied to solve bi-objective problems.

6. PREFERENCES

An e-learning system usually consists of electronic catalogue. An electronic catalogue is simply a table describing a set of items though their attributes. Users query the table in search of items of interest. However, more often than not, the answer set of a query contains several hundreds or thousands of items, most of which might be of no interest to the user. One solution to this problem is to present the items of the answer set in a decreasing order with respect to user preferences.

We call preference based query, or simply *preference query*, an ordinary query together with a set of preferences expressed by the user online, together with the query. Our main objective is twofold: (1) present an approach to the specification and evaluation of preference queries; (2) describe a user-friendly interface that allows a user to express a preference query over tabular data, and receive an answer set that is ordered in decreasing order of preference.

The specification and evaluation of preference queries has received considerable attention in the past several decades, mainly in the areas of decision support and databases. The use of preferences for ranking alternative choices has been around in decision making and social choice theory since the 1950s (see [33] for a comprehensive survey). However, the use of preferences in ranking query answers is quite recent and their embodiment in a query language presents a number of difficult problems (see [4], [13] and [21] for some influential papers). We note here that the use of preferences is also related to an important recent concept in databases, namely to the so called "skyline queries" [9].

Serial	Make	Color	Mileage	Price	Year
1	BMW	Black	35000	3800	2002
2	Honda	Blue	63000	2900	2000
...	...	...	...	...	...

Table 1: Table T, an e-catalogue for the sale of used cars (e.g. Autoreflex: http://pro.autoreflex.com/)

One issue related to preference queries is the kind of preferences that a user can express. The nature of preferences can be either quantitative or absolute (e.g. I like BMW 80%, I like VW 70%, ...) difficult to express by casual users but easy to compute, or qualitative (e.g. I like BMW more than VW) easy to express by casual users and easy to infer by a machine. The persistence in time of preferences can be long term, discovered by the system (unobtrusively, from query logs) or declared explicitly by the user and stored in the so called user's profile, or short term, expressed online explicitly by the user. The nature and persistence in time are orthogonal features of preferences. In this work we focus on short term, qualitative preferences (i.e. online preferences). In what follows, we sketch our approach through an example.

Consider the electronic catalogue T of an Internet company selling used cars (shown in Table 1). Each line of this table corresponds to a car (i.e. each tuple is the metadata record of a car), and the table might contain several thousands or even tens of thousands of tuples.

Consider now a usual query $Q = VW \vee BMW$ over T with preference $P = Red \rightarrow Black$ (meaning "Red is preferred to Black" as a color).

The system may output first a table T_{Red} containing all red cars which are either VW or BMW and then a table T_{Black} containing all black cars which are either VW or BMW.In that case, the user can "consume" the answer one "piece" at a time, in descending order of preference and stop "consumption" at any time.

One advantage of doing so is that the system can adapt to preferences. For example, if the preference Red $\rightarrow$ Black is replaced by Black $\rightarrow$ Red then T_{Black} will be shown first, followed by T_{Red}. To be able to support such an interaction with the user, we can follow one of the following two approaches:

1. First compute the answer to Q and then use the preference P to partition the answer set into the tables T_{Red} and T_{Black} to be shown to the user in that order.

2. First use the preference P to rewrite Q into two queries Q_{Red} and Q_{Black} so that their answers are the tables T_{Red} and T_{Black}.

The first approach is commonly used in the literature. In our work we propose to use the second approach, in the spirit of [19] [30].

7. CONCLUSION

In this paper, we presented data management in the GOD project. We introduced e-learning in the cloud. Then we

presented techniques to optimize data management, particularly caching strategies and rewriting rules, cost-based optimization using cloud-aware cost models and querying with preferences. In the near future, we will provide more details about the prototypes and results on the project Web site [31].

8. ACKNOWLEDGMENTS

Thanks to all the colleagues in CTU, ETIS, HELP University, LIMOS, LRI and University of Indonesia all the interesting discussions and for their extremely insightful remarks for improving this paper. This work is supported in part by the Ministère des Affaires étrangères and the CNRS (under grant STIC-Asie-12-GOD).

9. ADDITIONAL AUTHORS

Additional authors: Bruno Bachelet (Université Blaise Pascal, email: `bruno.bachelet@univ-bpclermont.fr`), Christophe Duhamel (Université Blaise Pascal, email: `christophe.duhamel@univ-bpclermont.fr`), Tao-Yuan Jen (Université de Cergy-Pontoise, email: `jen@u-cergy.fr`), Claudia Marinica (Université de Cergy-Pontoise, email: `claudia.marinica@u-cergy.fr`), Thuong Cang Phan (Université Blaise Pascal, email: `cang@univ-bpclermont.fr`), Gilbert Ooi Sin Cheak (HELP University, email: `gilbert.ooi@helpcat.edu.my`), Romain Perriot (Université Blaise Pascal, email: `romain.perriot@univ-bpclermont.fr`), Tran Thi To Quyen (Can Tho University, email: `tranthitoquyen@cit.ctu.edu.vn`) and Loïc Yon (Université Blaise Pascal, email: `loic.yon@univ-bpclermont.fr`).

10. REFERENCES

[1] Amazon. Dynamodb. web page. http://aws.amazon.com/dynamodb/.

[2] Amazon. Rds. web page. http://aws.amazon.com/rds/.

[3] Amazon. Simpledb. web page. http://aws.amazon.com/simpledb/.

[4] H. Andréka, M. Ryan, and P.-Y. Schobbens. Operators and laws for combining preference relations. *J. Log. Comput.*, 12(1):13–53, 2002.

[5] Apache. Hadoop. web page. http://hadoop.apache.org/.

[6] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and Others. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[7] Babson. Babson survey research group. Babson Survey Research Group.

[8] K. S. Beyer, V. Ercegovac, R. Gemulla, A. Balmin, M. Y. Eltabakh, C.-C. Kanne, F. Özcan, and E. J. Shekita. Jaql: A scripting language for large scale semistructured data analysis. *PVLDB*, 4(12):1272–1283, 2011.

[9] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, Heidelberg, Germany, 2001.

[10] R. Chaiken, B. Jenkins, P.-Å. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. Scope: easy and efficient parallel processing of massive data sets. *PVLDB*, 1(2):1265–1276, 2008.

[11] L. Chen, E. A. Rundensteiner, and S. Wang. Xcache: a semantic caching system for xml queries. In *SIGMOD*, page 618, Madison, Wisconsin, USA, 2002.

[12] B. Chidlovskii and U. M. Borghoff. Semantic caching of web queries. *VLDBJ*, 9(1):2–17, 2000.

[13] J. Chomicki. Preference formulas in relational queries. *ACM TODS*, 28(4):427–466, 2003.

[14] S. Dar, M. J. Franklin, B. T. Jonsson, D. Srivastava, and M. Tan. Semantic data caching and replacement. In *VLDB*, pages 330–341, Bombay, India, 1996.

[15] D. Dash, V. Kantere, and A. Ailamaki. An economic model for self-tuned cloud caching. In *ICDE*, pages 1687–1693, Shanghai, China, 2009.

[16] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, pages 137–150, San Francisco, California, USA, 2004.

[17] L. d'Orazio, C. Roncancio, and C. Labbé. Adaptable cache service and application to grid caching. *Concurrency and Computation: Practice and Experience*, 22(9):1118–1137, 2010.

[18] L. d'Orazio and M. K. Traore. Semantic cache for pervasive grids. In *IDEAS*, pages 227–233, Cetraro, Italy, 2009.

[19] P. Georgiadis, I. Kapantaidakis, V. Christophides, E. M. Nguer, and N. Spyratos. Efficient rewriting algorithms for preference queries. In *ICDE*, pages 1101–1110, Cancún, México, 2008.

[20] A. Y. Halevy. Answering queries using views: A survey. *VLDBJ*, 10(4):270–294, 2001.

[21] S. Holland and W. Kießling. Situated preferences and preference repositories for personalized database applications. In *ER*, pages 511–523, Shanghai, China, 2004.

[22] V. Kantere, D. Dash, G. Gratsias, and A. Ailamaki. Predicting cost amortization for query services. In *SIGMOD*, pages 325–336, Athens, Greece, 2011.

[23] A. M. Keller and J. Basu. A predicate-based caching scheme for client-server database architectures. *VLDBJ*, 5(1):35–47, 1996.

[24] D. Laurent and N. Spyratos. Rewriting aggregate queries using functional dependencies. In *MEDES*, pages 40–47, San Francisco, CA, USA, 2011.

[25] K. Lillis and E. Pitoura. Cooperative xpath caching. In *SIGMOD*, pages 327–338, Vancouver, BC, Canada, 2008.

[26] Microsoft. Sql azure. web page. http://www.windowsazure.com/en-us/home/features/data-management/.

[27] T.-V.-A. Nguyen, S. Bimonte, L. d'Orazio, and J. Darmont. Cost models for view materialization in the cloud. In *DanaC@EDBT*, pages 47–54, Berlin, Germany, 2012.

[28] C. Olston, B. Reed, U. Srivastava, R. Kumar, and A. Tomkins. Pig latin: a not-so-foreign language for data processing. In *SIGMOD*, pages 1099–1110, Vancouver, BC, Canada, 2008.

[29] N. Spyratos. The partition model: A deductive database model. *ACM TODS*, 12:1–37, 1987.

[30] N. Spyratos, T. Sugibuchi, and J. Yang. Personalizing queries over large data tables. In *ADBIS*, pages 271–284, Vienna, Austria, 2011.

[31] G. Team. God. web page. http://home.isima.fr/god/.

[32] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Z. 0002, S. Anthony, H. Liu, and R. Murthy. Hive - a petabyte scale data warehouse using hadoop. In *ICDE*, pages 996–1005, Long Beach, California, USA, 2010.

[33] A. Tsoukiàs. From decision theory to decision aiding methodology. *European Journal of Operational Research*, 187(1):138–161, 2008.

[34] P. Upadhyaya, M. Balazinska, and D. Suciu. How to price shared optimizations in the cloud. *PVLDB*, 5(6):562–573, 2012.

[35] A. Vancea, G. S. Machado, L. d'Orazio, and B. Stiller. Cooperative database caching within cloud environments. In *AIMS*, pages 14–25, Luxembourg, Luxembourg, 2012.