Designing Cloud Data Warehouses using Multiobjective Evolutionary Algorithms

Tansel Dökeroğlu[1], S. Alper Sert[1], M. Serkan Çinar[2] and Ahmet Coşar[1]
[1]*Department of Computer Engineering, Middle East Technical University, Ankara, Turkey*
[2]*Department of Computer Engineering, Hacettepe University, Ankara, Turkey*

Keywords: Cloud, Multiobjective Data Warehouse Design, Virtualization, Elasticity.

Abstract: DataBase as a Service (DBaaS) providers need to improve their existing capabilities in data management and balance the efficient usage of virtual resources to multi-users with varying needs. However, there is still no existing method that concerns both with the optimization of the total ownership price and the performance of the queries of a Cloud data warehouse by taking into account the alternative virtual resource allocation and query execution plans. Our proposed method tunes the virtual resources of a Cloud to a data warehouse system, whereas most of the previous studies used to tune the database/queries to a given static resource setting. We solve this important problem with an exact Branch and Bound algorithm and a robust Multiobjective Genetic Algorithm. Finally, through several experiments we conclude remarkable findings of the algorithms we propose.

1 INTRODUCTION

Cloud computing has emerged as a new computation paradigm that builds elastic and scalable software systems. Vendors such as Amazon, Google, Microsoft, and Salesforce offer several options for computing infrastructures, platforms software systems and supply highly-scalable database services with the goal of reducing the total cost of ownership price (Amazon, 2013). Users pay all costs associated with hosting and querying their data where service providers present different choices to trade-off price and performance to increase the satisfaction of the customers and optimize the overall performance (Balazinska et al., 2011). Recently, extensive academic and commercial research is being done to construct self-tuned, efficient, and resource-economic Cloud database services that protect the benefits of both the customers and the vendors. Virtualization that provides the illusion of infinite resources in many respects is the main enabling technology of Cloud computing. This skill is being exploited to simplify the administration of physical machines and accomplish efficient systems. The perception of hardware and software resources is decoupled from the actual implementation and the virtual resources perceived by applications are mapped to real physical resources. Through mapping virtual resources to physical ones as needed, the virtualization can be used by several databases that are located on physical servers to share and change the allocation of resources according to query workload requests. This capability of virtualization can provide efficient Cloud databases where each Virtual Machine (VM) has its own operating system and thinks that it is using dedicated resources (CPU, main memory, network bandwidth, etc.), whereas in reality the physical resources are shared among by using a VM Monitor (VMM) that controls the allocation of resources (Soror et al., 2010; Xiong et al., 2011; Curino et al., 2011).

In addition to providing efficient queries in accordance with the service level agreements, contemporary relational Cloud database management systems need to optimize a multicriteria problem that the overall cost of hardware ownership price is also to be minimized. In this study, we develop a framework to provide (near-)optimal virtual resource allocations with respect to the overall cost of hardware ownership price and a good tradeoff between the efficiency and the overall cost of a relational data warehouse is ensured. More specifically 'Given a budget constraint and a query workload, how can the available virtual resources of

Dökeroğlu T., Alper Sert S., Serkan Çinar M. and Coşar A..
Designing Cloud Data Warehouses using Multiobjective Evolutionary Algorithms.
DOI: 10.5220/0004906805710576
In *Proceedings of the 6th International Conference on Agents and Artificial Intelligence* (ICAART-2014), pages 571-576
ISBN: 978-989-758-015-4

the Cloud (CPU, main memory, network bandwidth, I/O bandwidth, and etc.) be allocated to Virtual Machines, each having a part of a distributed database that the best overall query performance can be achieved with minimum pricing?' Our framework produces cost-efficient design alternatives (virtual resource configuration and query plans) and recommends them to the decision makers. A budgetary constraint can be a more important issue for a consumer, whereas the response time of the queries is more crucial for another (D'Orazio et al., 2012). Therefore, in order to fully realize the potential of the Cloud, alternative query plans are executed with well configured virtual resources instead of only optimizing single query plans on statically designed virtual resources. This means that instead of designing the database over standard VMs, we configure the virtual resources. It indicates that CPU usage and RAM can be crucial for a data warehouse workload, whereas network or I/O bandwidth can be more important for another.

In order to explain the multiobjective query optimization problem, we give an illustrative example. Consider a distributed decision support database (TPC-H) where all of its eight tables are located on different VMs. When we execute TPC-H Query 3 with two different query plans and with alternative virtual resource allocations, we certainly obtain different performances. During the experiments, the configuration with higher performance VMs (4 x 2Ghz CPU, 8GB RAM, 300Mbit/sec network bandwidth) and with query plan QP1 SQL statement is observed to be the fastest performing platform, however its monetary price is one of the most expensive alternatives. A cheaper VM configuration with 1 x 2Ghz CPU, 768MB RAM, 100 Mbit/sec network bandwidth and with QP1 SQL statement has a response time that is only 25.9% slower but 72.0% cheaper. The pareto-optimal visualization of the proposed solutions that we obtain by executing with different VM and network configurations is presented in Figure 1 (see Appendix for query plans). Looking at the results we can see that finding solutions that are closer to the ideal point is the main objective of the problem. In a two dimensional search space, the optimizers explore solutions for this NP-Hard problem. Solutions with appropriate VMs, network bandwidth and efficient QP configurations can meet the demand of consumers.

Some queries may request more network resource, whereas others consume more CPU and main memory. In this case, the former one can be executed in a better way by spending the budget on a broader network bandwidth instead of multiple CPUs. In order to investigate the effectiveness of our approach, we incorporate the devised framework into a prototype system for evaluation and instantiate it with an exact solution method, multiobjective branch-and-bound and an evolutionary computation method multiobjective genetic algorithms. Finally, through several experiments that we conduct with the prototype elastic virtual resource deployment optimizer on different TPC-H query workloads, we conclude significant results of the space of alternative deployments as well as the advantages and disadvantages of the multiobjective optimization algorithms.

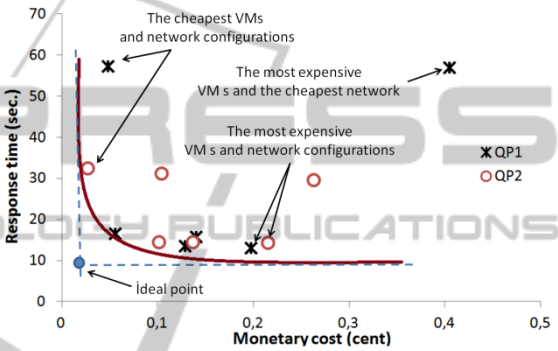

Figure 1: Pareto-optimal curve for the response time and monetary cost of TPC-H Q3 with different virtual resource configurations and query plans.

3 MULTIOBJECTIVE DATA WAREHOUSE DESIGN AND EXPERIMENTS

The studies concerning the performance of multiobjective Cloud databases are at their early ages. Most of the distributed database design and optimization concepts can be applied to this area however; multiobjective optimizations are newly being studied. There has been substantial amount of work on the problem of tuning database system configurations for specific workloads or execution environments (Weikum and Vossen, 2002) and on the problem of making database systems more flexible and adaptive in their use of computing resources. On the other hand, to the best of our knowledge, there is no approach like ours that concern both with the optimization of the total ownership price and the performance of the queries of a data warehouse by taking into account alternative virtual resource allocation, and different query plans workloads, and materialized views.

In this study, we tune the virtual resources to a distributed data warehouse system, rather than tuning the database system for a given resource setting. Moreover, our study optimizes the objectives of minimum money consumption and maximum benefit from the virtual resources by using multiobjective branch and bound genetic algorithms. In summary, our study focuses on the elasticity of Cloud resources and produces multiple virtual resource deployment plans with alternative query plans for a set of queries in a workload, enabling the user to select the desired tradeoff with efficient cost models. Materialized views are effective techniques for speeding up query workloads and they are increasingly being used by many commercial database systems. Materialized views are especially good for data warehouses because of the intensive usage of common subexpressions including select-project-join operations. In our study, we focus on the selection of the appropriate materialized views to reduce the communication cost, response time, and the ownership price of a relational Cloud database with respect to the pricing scheme of vendors. The multiobjective query optimization can benefit from carefully selected materialized views. In addition to reducing the response time of the queries, total ownership prices decrease significantly. Although storing/maintaining the selected materialized view has an additional cost, it is still a very effective way of executing queries. No resource deployment processing system deals with the concept of elasticity and cost-efficiency of relational Cloud databases that makes use of the appropriate materialized views like our system.

Our design system relies on multiobjective query optimizers. Therefore, we turn a conventional query optimizer into a multiobjective query optimizer. During the execution of the queries, we use a variant of operator centric model that is known to be more appropriate than the classical iterative-based (pull-based) query execution models for multiple query optimizations. Main component of this system is an alternative query generator that generates different bushy-plan based query plans for the incoming queries.

In order to design a multiobjective Cloud data warehouse, a multiobjective query optimizer must be used. Therefore first thing we do is to enhance a conventional query optimizer to a multobjective one.

Data storage cost depends on the size of the data (including the structures such as indexes, materialized views, and replications of the tables) and its storage period. Processing time of the VMs is the total price for CPU usage. During the execution of the queries, different VM configurations can be used and the configuration of a VM (RAM, number of CPUs, and etc.) is flexible in accordance with the resources used. Micro, small, large, and extra large are some of the configurations provided by the Cloud vendors at various prices. Data transfer cost is related with the amount of data migrated between sites.

Alternative query plans (QP) provide different ways for executing a query. Alternative QPs can take advantage of different ways of executing the same query, thus cheaper resources can reduce the total price of a query while increasing the response time. This elasticity provides new opportunities for the solution of our multiobjective problem.

The formulation of the problem consists of two parts. The monetary cost and the response time of the query workloads that will work on the selected VMs with the alternative QPs of the queries. There are n VMs with independent DBMS and each VM has a set of processors and a main memory. Each DBMS has a workload that consists of a set of SQL statements. Workload represents the queries submitted to DBMS i. There are m different physical resources (CPU capacity, main memory, network bandwidth, etc.) that are to be deployed to VMs. Our main goal is to obtain a set of pareto- optimal solutions that the overall monetary and response time cost are minimized.

Pricing Scheme Parameters of the Cloud: Each customer requests queries from the Cloud data warehouse by using Internet and contacts with the aggregate node. The aggregate node distributes the query to the appropriate VM. The Cloud infrastructure provides unlimited amount of storage space, CPU nodes, RAM, and very high speed intra-cloud networking. All the resources of the Cloud are assumed to be on a network. The CPU nodes, RAM, and I/O bandwidth of each VM are different from each other and can be deployed by using VM Monitors in milliseconds (Barham et al., 2003). The storage system is based on a clustered file system where the disk blocks are stored close the CPU nodes accessing them. I/O bandwidth of the storage is divided evenly to the VMs (that may have multiple cores up to 8).

There are several Cloud Service Providers (CSP) in the market and they offer different pricing schema for the services they provide. Different pricing schema of Cloud server providers can be opportunities for customers in accordance with the tasks they want to complete. The cost for a small VM (1GHz CPU, 768MB RAM) is $0.02/hr,

whereas A7 (8 x 1.6GHz CPU, 56GB RAM) is $1.64/hr. Data storage is also billed by the Cloud service providers. In our model, monthly storage price is used. During our experiments, the data storage price is constant for all the queries. But for database designs that use materialized views an additional storage cost needs to be added. Most of the Cloud providers do not charge for the data transfers in a private Cloud but the data that leaves the Cloud. The bandwidth of the intra-Cloud network can reach up to 10GBit/sec. In order to make our problem more interesting and handle this dimension of the optimization, we have located our VMs on a virtual switch. Different bandwidth networks can be chosen and the pricing scheme changes in this communication infrastructure.

Branch-and-Bound Algorithm (MOD-B&B) is an exhaustive optimization algorithm. It enumerates all candidate solutions, where fruitless subsets of candidates are discarded, by using upper and lower estimated bounds of the problem instance being optimized. MOD-B&B starts searching with null initial values indicating that no QP has yet been selected for any queries with the current VM configuration. Later, QPs are assigned to current selected VM. At each level of the tree, one additional QP is assigned to the query workload (Bayir et al., 2007). This procedure is repeated for every VM configuration. We define two initial upper bounds for MOD-B&B. The minimum monetary cost is the running time of VMs that execute the queries in a workload of queries. The response time is the finishing time of the workload with the given VM configuration. In order to estimate a lower bound, different heuristic functions can be used. The heuristic we proposed here is reasonable and performs well during the optimization process. We will explain the heuristic with a scenario. In Figure 2, we can see the results of a sample multiobjective query workload optimization. The best response time and the minimum monetary cost values are defined and marked on the Figure. We can obtain these values with the most expensive and the cheapest VM configurations easily. Hereby, we propose a heuristic point (marked as Heuristic point on the Figure) that is the center of the square constructed by the response time and monetary costs of the best and worst (the cheapest) VM configurations. If the response time of a workload falls above heuristic point or if the monetary cost is at the right-hand side of heuristic point on the Figure then it is pruned according to our heuristic.

Multiobjective Genetic Algorithm (MOD-GA): The principles of applying natural evolution to

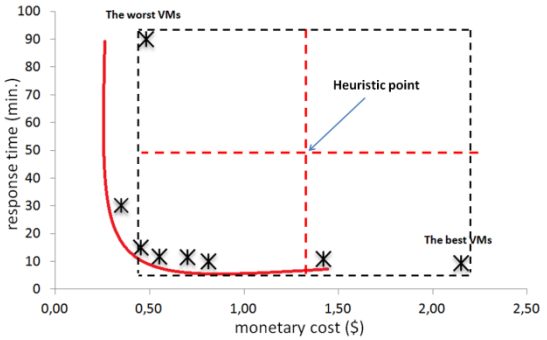

Figure 2: Proposed heuristic value for MOD-B&B algorithm.

optimization problems were first described by Holland (1975). The GA theory has been further developed and GAs have become very powerful tools for solving search and optimization problems (Tosun et al., 2013; Dokeroglu, 2012). GAs are based on the principle of genetics and have been frequently used to solve many NP-Complete problems. GAs use a computational model that simulates the natural processes of selection and evolution. Individuals with better quality have more chance to survive, to reproduce, and to pass their genetic characteristics to future generations. Each potential solution in the search space is considered as an individual and is represented by strings called chromosomes. Genes are the atomic parts of chromosomes and codify a specific characteristic. Chromosomes are encoded in different ways for each application. A random population is generated in the first step of the algorithm and by applying selection, crossover, and mutation operations iteratively, new generations are created. The individual having the best fitness value in the population is returned as the solution of the problem. Our multiobjective data warehouse design problem can be modeled by using evolutionary methods. A chromosome corresponds to a solution instance including a set of VMs (having different CPU, RAM, I/O bandwidth and etc.) with tables/replications located on their databases, alternative query plans (QPs) of queries in the workload, and a network gene. Figure 3 shows the chromosome structure of a solution. The leftmost segment represents the configuration of the VMs with the tables/replications on its database. Middle segment is the set of QPs for the queries in the workload. Rightmost part gene represents the selected network layer of the solution vector. Because we do not make use of partitioned fragments of the relations, the size of the VM segment of the chromosome can be at most as many

as the number of tables in the database. Using fragments (partitions) of larger tables can even further improve the performance as it is used by most of the commercial data warehouses. Our proposed genetic model can be enhanced to include the fragments of the tables.

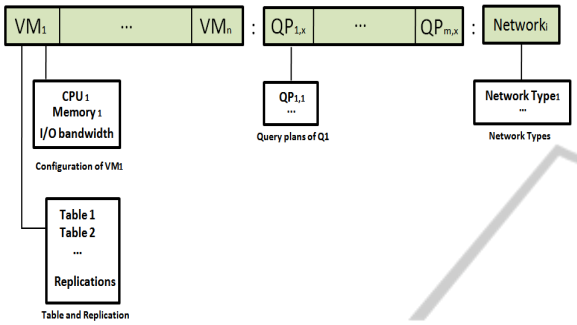

Figure 3: Chromosome structure for the proposed multiobjective genetic algorithm that consists of Virtual Machines, query plans and a network layer.

We have introduced three operators for the MOD-GA. Global and local crossovers, and a mutation operator. Global crossover operator uses two parents that are selected from the population by a selection method. We have proposed two types of crossover operators, global and local. Global crossover operator swaps VM, QP, or network part of two selected chromosomes with the same counter chromosome. Below we can see two parents and their VM parts are exchanged to provide two new chromosomes. Local crossover operator works on the VM and QP segments of the chromosome by dividing the parents and exchanging the segments with each other. Mutation operator changes a randomly selected gene of a chromosome. In our chromosome structure this operator can act on any of the segments. Only a gene is replaced at every mutation process.

Experimental Results: In this part, we perform experiments with a sample TPC-H workload. The workload is first optimized with MOD-B&B and MOD-GA algorithms. Later, selected solutions that are produced by our algorithms are executed in our Cloud Database environment to verify the correctness of our approach. There are alternative selected set of VM and query plan configurations in these results. The solutions are used to measure the effectiveness of other solutions. The workloads are executed 10 times with the selected VM configurations and average values are presented. In Figure 4, we present the results of pareto-optimal solutions that are produced by MOD-B&B (represented with **+** sign), MOD-GA (represented with **x** sign) and solutions with average prices (represented with triangle sign). The solutions with the highest and the cheapest performance VMs are also added to define upper and lower bounds. VMs with the highest configuration capabilities (XL) give the best response time and WMs with the worst configurations (XS) give the longest execution time. In this sense, they provide meaningful results to evaluate the quality of solutions provided by MOD-B&B, and MOD-GA. In the Figure, a hypothetical ideal point is defined to show the optimal fitness value that can be achieved within the given minimum response time and minimum pricing. The solutions that are chosen from the set of solutions produced by MOD-B&B, and MOD-GA algorithms construct a pareto-optimal convex curve that a decision maker can choose any of the solutions according to his/her requirements. The MOST expensive VMs option gives the fastest response time and the cheapest VMs option is the most time consuming.

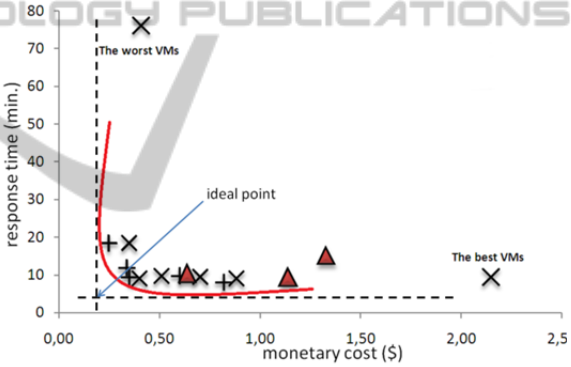

Figure 4: Proposed pareto-optimal solutions for a sample TPC-H workload with MOD-B&B and MOD-GA algorithms.

Although view materialization has additional storage costs for a Cloud data warehouse, it provides faster response times up to 60-80% for query workloads. It is observed to be an efficient way of designing a Cloud data warehouse with respect to monetary and response time costs. The experimental results concerning the appropriate selection of the materialized views for the Cloud data warehouses are not presented due to the space limitations.

4 CONCLUSIONS

In this paper, we define some principles to design efficient multiobjective Cloud data warehouses by making use of the elasticity of the virtual resources.

We minimize the monetary cost as well as providing fast response times. We formulate the problem and propose exhaustive and heuristic algorithms, namely, multiobjective branch-and-bound (MOD-B&B) and multiobjective robust genetic algorithm (MOD-GA) for the optimization of the problem. To the best of our knowledge, the multiobjective design of Cloud data warehouses is being solved for the first time with such an approach. There are studies that concern with the best virtual resource deployment or with the minimal monetary cost of workloads in static hardware resources individually. However we combine both of these optimization techniques together and obtain significant results as they are presented in our study. It is possible to design and expand the study with additional elastic virtual resources such as I/O bandwidth and dynamic RAMs.

REFERENCES

Amazon Web Services (AWS). aws.amazon.com.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., and Warfield, A. (2003). Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review, 37(5),* 164-177.

Balazinska, M., Howe, B., and Suciu, D. (2011). Data markets in the cloud: An opportunity for the database community. *PVLDB, 4(12).*

Bayir, M. A., Toroslu, I. H., and Cosar, A. (2007). Genetic Algorithm for the Multiple- Query Optimization Problem. *IEEE Transactions on Systems, Man, and Cybernetics- Part C: Applications and Reviews, Vol. 37 (1)*:147-153.

Curino, C., Jones, E., Popa, R., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., and Zeldovich, N. (2011). *Relational Cloud: A Database Service for the Cloud.* CIDR, pp.235-240.

D'Orazio, L., Bimonte, S., and Darmont, J. (2012). Cost Models for View Materialization in the Cloud. *In Proceedings of the Workshop on Data Analytics in the Cloud (EDBT-ICDT/DanaC).*

Dokeroglu, T. (supervised by Ahmet Cosar) (2012). Parallel Genetic Algorithms for the Optimization of Multi-Way Chain Join Queries of Distributed Databases, *38th VLDB Ph.D. Workshop,* August 27-31, Istanbul/TURKEY.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA.

Soror, A. A., Minhas, U. F., Aboulnaga, A., Salem, K., Kokosielis, P., and Kamath, S. (2010). Automatic virtual machine configuration for database workloads. *ACM Transactions on Database Systems (TODS)*, 35(1), 7.

Tosun, U., Dokeroglu, T., and Cosar, A. (2013). A robust Island Parallel Genetic Algorithm for the Quadratic Assignment Problem. *International Journal of Production Research*, 1-17.

Weikum, G. and Vossen, G. (2002). *Transactional Information Systems. Morgan Kaufmann*.

Xiong, P., Chi, Y., Zhu, S., Moon, H. J., Pu, C., and Hacigumus, H. (2011). Intelligent management of virtualized resources for database systems in cloud environment. *In Data Engineering (ICDE), IEEE 27th International Conference on* (pp. 87-98).

APPENDIX

Alternative query execution plans for TPC-H query Q3

TPC-H Q3 statement in accordance with the query execution plan 1 where all of the tables are shipped to query issuing node.

```
SELECT TOP 10 L ORDERKEY ,..., O SHIPPRIORITY
FROM [VM2].CUSTOMER C, [VM3].ORDERS O,
[VM1].LINEITEM L,
WHERE C.C MKTSEGMENT = 'BUILDING'
AND C.C CUSTKEY = O.O CUSTKEY
AND L.L ORDERKEY = O.O ORDERKEY
AND O.O ORDERDATE < '1995-03-15'
AND L.L SHIPDATE > '1995-03-15'
GROUP BY L.L ORDERKEY, O.O ORDERDATE, O.O
SHIPPRIORITY
ORDER BY REVENUE DESC, O.O ORDERDATE;
```

TPC-H Q3 statement in accordance with query execution plan 2 where CUSTOMER and ORDERS tables are joined at virtual machine 3 and the resulting tuples are shipped to virtual machine 2 to join with LINEITEM table.

```
SELECT TOP 10 L ORDERKEY ,..., O SHIPPRIORITY
FROM OPENQUERY ( [VM3]. 'SELECT O ORDERDATE, O
SHIPPRIORITY, O ORDERKEY
FROM [VM2].CUSTOMER C, [VM3].ORDERS O
WHERE C.C MKTSEGMENT = 'BUILDING'
AND C.C CUSTKEY = O.O CUSTKEY
AND O.O ORDERDATE < '1995-03-15'
GROUP BY O.O ORDERDATE, O.O SHIPPRIORITY, O
ORDERKEY ORDER BY O.OORDERDATE) Remote1,
[VM1].LINEITEM L WHERE AND L.L ORDERKEY =
Remote1.O ORDERKEY AND L.L SHIPDATE > '1995-03-15'
GROUP BY L.L ORDERKEY, Remote1.O ORDERDATE,
Remote1.O SHIPPRIORITY
ORDER BY REVENEU DESC');
```