2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing

A Heuristic Query Optimization Approach for Heterogeneous Environments

Peter Paul Beran, Werner Mach, Ralph Vigne, Jürgen Mangler and Erich Schikuta
University of Vienna
Department of Knowledge and Business Engineering
Rathausstr. 19/9, A-1010 Vienna, Austria
Email: {peter.beran,werner.mach,ralph.vigne,juergen.mangler,erich.schikuta}@univie.ac.at

***Abstract*—In a rapidly growing digital world the ability to discover, query and access data efficiently is one of the major challenges we are struggling today. Google has done a tremendous job by enabling casual users to easily and efficiently search for Web documents of interest. However, a comparable mechanism to query data stocks located in distributed databases is not available yet. Therefore our research focuses on the query optimization of distributed database queries, considering a huge variety on different infrastructures and algorithms. This paper introduces a novel heuristic query optimization approach based on a multi-layered blackboard mechanism. Moreover, a short evaluation scenario proofs our investigations that even small changes in the structure of a query execution tree (QET) can lead to significant performance improvements.**

***Keywords*-heuristic A*-algorithm, cost-based query optimization, multi-layered blackboard system, Service-Oriented Database Architecture**

I. Introduction

In the last decades [1] inexpensive resources, such as processor, memory and hard disk, yield to innovative inventions in the database sector. A growing database demand together with rapidly emerging technologies such as multiprocessors force the development of sophisticated parallel database systems. Hereby, parallelism is usually achieved by facilitating intra- or inter-operator parallelism in a network of database machines that are bundled together, often referred as federated databases.

In this paper we present a novel multi-layered blackboard architecture for the heuristic optimization of parallel query execution plans in heterogeneous environments. Generally the performance of such distributed queries heavily depends on the quality of the constructed QEP. That is to say the performance of the used relational operations (e.g. sort, join) and their optimal sequence performed on data sets derived from a multitude of distributed databases. In our solution we denote how to choose the right data sources, elect appropriate execution resources and extend algebraic transformation rules to perform a query (near) optimal. Moreover an evaluation model proofs our observations that even small modifications in the structure of a QEP lead to benefits in the query optimization.

II. Blackboard Query Optimization

The elementary idea behind a blackboard system is to solve NP-hard problems heuristically, like a group of human experts would do when they gather around a common shared space (blackboard) and place their ideas concerning a specific problem. By iteratively and repetitively re-formulating and altering the available ideas it is possible to solve a given problem and find an appropriate solution. This traditional notion of a blackboard was originally introduced by the Artificial Intelligence [2] society consisting of the following three components:

1) A **global blackboard** that is usually represented by a conventional database sharing information about input data and partial solutions.
2) A **knowledge base** (regions) as a storage facility containing the expert knowledge in independent units, often referred as regions. Hereby, each region is owned by a single expert and communication between regions is accomplished using the global blackboard.
3) A **control component** (phases) dictates the course of activities for the problem solving approach. In order to make reasonable decisions each region has to provide cost estimations for their applied operations to build up a cost-based decision tree.

In our approach this technique is used in the domain of distributed databases and has been adapted to find (near) optimal QEPs within heterogeneous database environments. Moreover, the problem domain has been divided into three layers (Fig. 1). Each layer is realized using a blackboard targeting a specific problem aspect by iteratively processing its optimization (local iteration). All blackboards use an A^*-algorithm offering possibilities to prune the search space and thus avoid exhaustive searching. This allows dynamically reacting to changing conditions and concentrating only on promising solutions. The decision tree consists of nodes, which resemble partial solutions to the problem and their estimated costs. In a step-wise approach the algorithm starts at the root and expands nodes whenever the estimated costs associated with that node are lower than the costs of any given solution that has already been found. Solutions which are heuristically known or proven to be inefficient are discarded automatically. After completion of a layer

978-0-7695-4039-9/10 $26.00 © 2010 IEEE
DOI 10.1109/CCGRID.2010.65

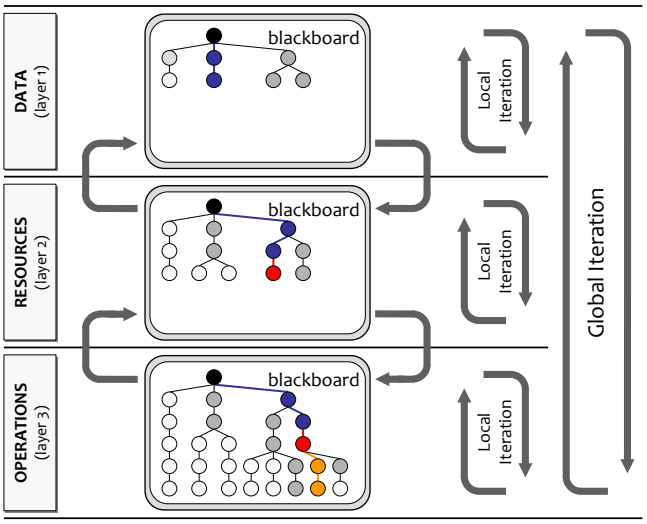


Figure 1. Multi-layered Blackboard

the process continues with the next layer. However it is possible to step back one layer if another solution path, due to changing conditions, now possesses the minimal costs (global iteration).

Generally two different parameter types used for the nodes of the decision tree can be identified.

- **Continuous parameters** denote parameters which are either to be minimized or maximized regarding a cost function. The value domain is usually limited by a border value that denotes the least acceptable value for the execution in the QEP. Values beyond the border are not acceptable and result in ∞-costs.
- **Boolean parameters** describe parameters with the value domain "true" or "false". Met parameters have no costs, otherwise ∞-costs.

Additionally we introduce the concept of priorities that allow to weight parameters according to their importance. Priority values can range from 0 (no importance) to 1 (mandatory). The multilayered blackboard consists of three layers, each a blackboard on its own:

A. Data Blackboard

It contains all data-specific issues concerning data- locality, distribution, movement, replication and partitioning. Rules related to vertical and horizontal partitioning or coping with data skew are applied here. Therefore metadata information must be available for the existing data sources. Referring to our preceding research [3] a profound analytical model for operations covering the most important data aspects (such as data locality, network bandwidth) has been developed. One main result was that varying network bandwidths may cause the transmission time for input and output data to change dramatically.

B. Resources Blackboard

It concentrates on the assignment and allocation of computing resources. Thus it benefits from the optimization capabilities for homogenous and heterogeneous environments, considering time-invariant issues (e.g. network link bandwidth, CPU frequency) as well as time-variant issues (e.g. actual network bandwidth or CPU load). It also handles application dependent issues that are bound to the profile of the executed application (e.g. access patterns) and non-technical issues (e.g. infrastructure costs, political regulations). Due to the use of agents a service provider, and so forth the resources blackboard, is able to negotiate QoS (quality of service) parameters regarding resource-specific requirements. Using our SODA [4] framework, a SOA-enabled prototype of a distributed database system, we successfully demonstrated how such a resource classification and characterization can be applied for heterogeneous distributed resources.

C. Operations Blackboard

It covers the query optimization itself and consists of many consecutive regions, each containing a sequence of expansion rules to transform an existing QET reflecting a certain status of the optimized query. At the beginning a user query is translated into a normal form similar to a most costly normal form [5] building up the starting-point for the optimizer. To propagate the items through the regions the transformation rules of the knowledge base are applied. These rules are generally based on the rules for traditional database query optimization and are grouped according to the database operation type: select, project, join, union, difference, complement, and semi-join. Other operations like Cartesian product, sort (order by), group, aggregate (count, sum, max, min, avg) and intersect can be composed out of the given ones. The term "pushing up/down" denotes the transformation of the QEP-tree by moving operators up or down in the tree structure. However all transformation rules only affect the syntactic notation of the QEP, which in turn changes the execution sequence of the query, but does not affect the query result.

III. Analysis and Evaluation

The proposed evaluation scenario illustrates our approach and clarifies how we measure the enhancements of our multi-layered blackboard architecture. An exemplary query is processed to obtain a (near) optimal query execution plan (QEP), concerning data-, resource- and operation-specific parameters and looks like the following expressed in Structured Query Language (SQL):

```
SELECT * FROM part ORDER BY p_size
```

The chosen database model was derived from the TPC-H benchmark. For our scenario we only cover the `part`

table, selecting all attributes (columns) and sorting them using their `size` attribute in an ascending order. Thus the challenge is to choose the best data location and the fastest sort algorithm. Furthermore all parameters of the blackboard layers are assumed to be "mandatory" (priority = 1.0).

A. Evaluation Environment

In our evaluation we use SODA (Service Oriented Database Architecture) [4]. It consists of small building blocks – implemented as Web services – which can be plugged together almost in a LEGO™ like manner to shape a QET. It provides the business logic for a distributed query execution, specifically supporting autonomous databases in heterogeneous environments. Each Web service implements an atomic database operation such as projection, selection, sorting, join or Cartesian product. Furthermore operations exist to retrieve (fetch) and store (sink) tuples used within a query execution process.

B. Query Optimization

The initial blackboard is depicted in Fig. 2(a) whereas each possible parameter value is depicted by a node (rectangle) in the corresponding layer. Edges between nodes indicate logical implications, e.g. if data sources with a global schema are used then the replication degree can be either 2 or 5. Between different layers no edges are drawn because here every combination is possible, e.g. it does not matter which replication degree was chosen to select the processors parameter (varying from 1 to 16 cores). The exponent next to the bandwidth states how many nodes achieve the bandwidth. Using the A^*-algorithm of a blackboard allows us to select the required data sources (layer 1), choose sufficient computational resources (layer 2) and incorporate the most appropriate operations (layer 3) based on a cost-based estimation.

From this starting point the query optimization follows an iterative and incremental approach, where in each loop a the operator with the lowest costs is expanded and possibly new sub-trees are exposed.

Data Blackboard: Concerning the data issues the *Data Schema* and the *Replication Degree* parameters have to be optimized. As in the scenario the provision of a global schema is compulsory, the right node is chosen (costs: 1.0). All nodes providing only a local schema are assigned with ∞ costs and are so forth not further considered in the optimization. Due to the fact that a data copy is needed, the replication degree border value is set to 1. Thus the node with replication degree 5 (costs: 0.33) is selected instead of the node with replication degree 2 (costs: 0.66), a typical maximization problem.

Resources Blackboard: Concerning the resource issues the number of available *Processors* (static parameter) and the network *Bandwidth* (dynamic parameter) of the processing nodes have to be maximized. For the computational resources a lower border of two processors is mandatory, while the network bandwidth of each processor must not under-run 5Mbit/s. If only one processor is available, or the bandwidth of at least one node drops off below 5Mbit/s the costs are ∞. In the first attempt the right-most node (costs: 0.18) providing 16 cores is selected. But the next node does not fulfill the bandwidth criteria of at least 5Mbit/s per processor, so the costs become infinite. Due to this fact the decisions has to be revised, switching to the configuration with only 8 cores (Fig. 2(b)). The costs of the two sub sequential nodes are computed and the left successor node (costs: 0.75) is chosen, each processor providing a bandwidth of 7Mbit/s. The other possibility is – at the moment – not further investigated (costs: 0.80).

Operations Blackboard: Concerning the operation issues only basic cost functions for relational operations have to be considered. First of all the SQL-query is translated to its corresponding normal form. Based on this normal form the QET can be created consisting of a π projection and $\Downarrow\Uparrow$ sort operation layer only. Implicitly δ declustering and $\cup$ union are facilitated to support the declustering operations allowing i.e. for total parallelization. The optimizer applies one π-operator and three different $\Downarrow\Uparrow$-operators ($\Downarrow\Uparrow_q$ for quick-sort, $\Downarrow\Uparrow_{pbb}$ for parallel block bitonic-sort and $\Downarrow\Uparrow_{pbm}$ for parallel binary merge-sort). For the expansion of the π-operation (costs: 1.00) there is no other choice, so it is selected. In a first attempt the $\Downarrow\Uparrow_{pbb}$ (costs: 1.00) is chosen, instead of the $\Downarrow\Uparrow_{pbm}$ (costs: 1.01) because it is cheaper. Due to a rule for a modified $\Downarrow\Uparrow_{pbm}$ that benefits from a single faster node in its postoptimal phase [6] the costs can be reduced to 0.94. So the overall solution is 0.01 cheaper than the former one using the $\Downarrow\Uparrow_{pbb}$.

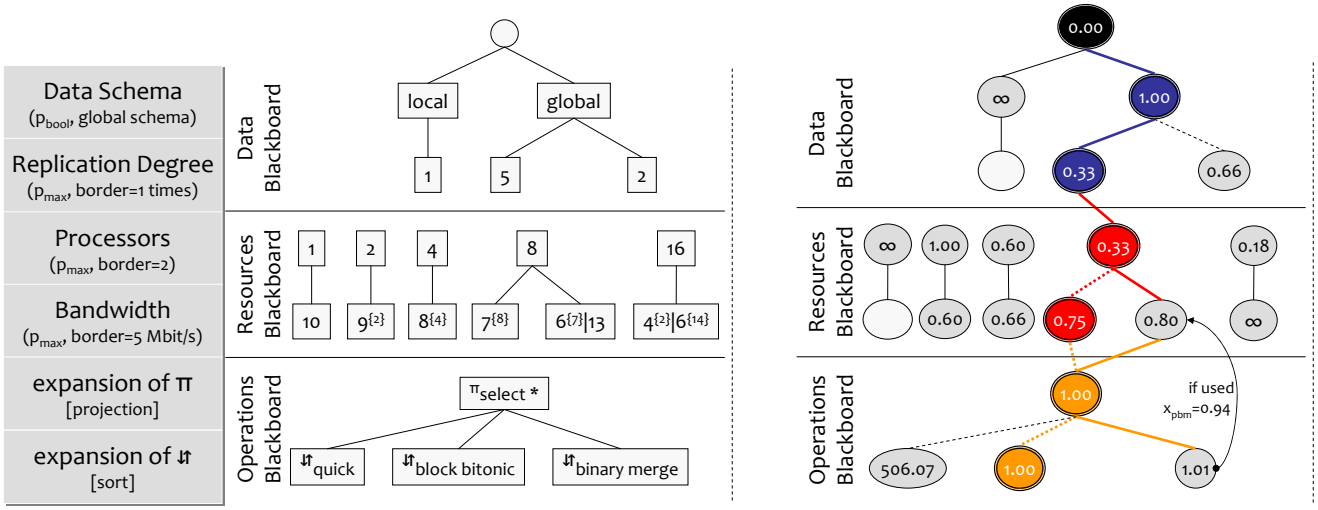
Finally the costs of the blackboard estimation are 4.40, according to Fig. 2(b). The solid bold line indicates the optimal path of decisions that have to be taken to perform the query in an optimal way.

C. A Modified Sort Model for Heterogeneous Environments

Based on the work of Bitton et al. in a generalized multiprocessor organization the block bitonic sort has in any case a better performance then the binary merge sort, based on the analysis in [1]. To analyze the mapping of the algorithms onto a simplified heterogeneous environment we have to specifically pay attention to the last phase (postoptimal) of the three phases of the algorithms (Formula 1).

$$\underbrace{\frac{n}{2p}log(\frac{n}{2p})}_{suboptimal} \quad \underbrace{+\frac{n}{2p}}_{optimal} \quad \underbrace{+logp - 1 + \frac{n}{2}}_{postoptimal} \tag{1}$$

The last phase (postoptimal) of the binary merge sort algorithm is split into two parts (Formula 2) because only



(a) Initial Blackboard with possible Choices

(b) Final Blackboard with optimal Solution

Figure 2. Blackboard for Evaluation Scenario

one processor is necessary doing the merge for $\frac{n}{2}$ costs.

$$\underbrace{[logp - 1 + \frac{n}{2}]}_{postoptimal} \rightarrow \underbrace{logp - 1}_{postoptimal_I} \underbrace{+\frac{n}{2}}_{postoptimal_{II}} \quad (2)$$

If we choose for this "bottleneck" the nodes of the heterogeneous environment (Grid or respectively Cloud) with the best network bandwidth available, the effect on the overall performance has to be at least noticeable. We specifically emphasize that even only one processing node with high network performance is worthwhile to exploit this effect. This leads to the clear policy for orchestration of a workflow with participating heterogeneous resources for a parallel binary merge sort to use nodes with the highest network performance in the $postoptimal_{II}$ phase as laid out in algorithm. Vice versa using one high performance node in the bitonic sort gives no performance gain at all because this node is slowed down by all other nodes working in parallel in a staged manner. This effect that the binary merge sort now outperforms the block bitonic sort in a simplified heterogeneous organization is shown in [6]. It can be easily explained by Amdahl's law too, where the performance of a parallel algorithm is dependent on its sequential part. More specific, Amdahls's law is stating that the speedup is limited by the sequential part. It is clear that even the most powerful node will limit the performance increase if the number of used nodes for the whole parallel algorithm is increasing. A speedup and scale-up analysis [4] for the used SODA environment also verifies this issue. Please note that we had used a heterogeneous environment for our performance analysis and not a homogeneous environment like common sort benchmarks (e.g. Terasort[1]), that means a comparison to these benchmarks does not make sense.

[1] http://sortbenchmark.org/

D. Proof of the Modified Sort Model

To proof our statements of the preceding sort model we implemented the discussed algorithms using SODA. The performance measurement of our algorithm is related to a derived version of the TPC Benchmark™H, only adopting the `PATH` table (as pointed out in the scenario) and using the *DBGEN* tool to populate the database. The performance analysis by the analytical model in [6] shows that a smart orchestration of the available nodes with heterogeneous performance characteristics, results in an increased performance behavior, as depicted in Fig. 3.

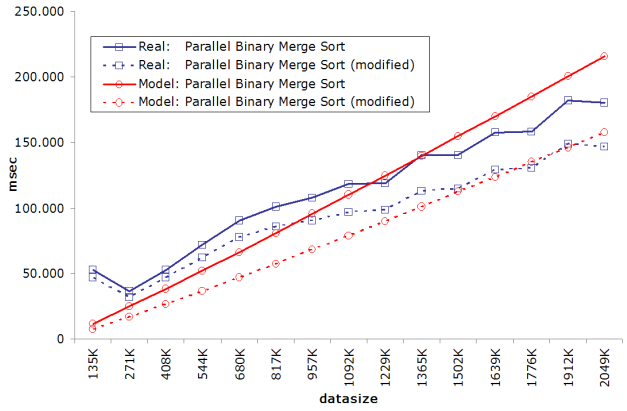


Figure 3. Real and Model performance behavior of Parallel Merge Sort, unmodified and modified

For the practical implementation a heterogeneous hardware infrastructure consisting of a mix of three different

blade systems (with a varying number and combination of cores) was used. For the measurements the workflow was executed in an unmodified and a modified style. In case of the modified version the network speed of the last three nodes (two merge nodes sending their data to the final merge node) was fixed to 1GBit/s. Apart from that during various runs the network connection speed has been adjusted between 512Kbit/s up to 1GBit/s to simulate a typical heterogeneous environment and proof our hypothesis.

According to our presented orchestration algorithm we placed the postoptimal phase of the binary merge sort algorithm on the node with the highest performance and best bandwidth connection. The practical results justified our analysis that the most influencing factors are the processor performance and the network bandwidth. Fig. 3 shows the actual execution times of the parallel merge sort algorithm for a conventional scenario and the modified one according to our orchestration algorithm. The expected performance improvement is easily recognizable and proves our analytical findings. The benchmark configuration uses 8 nodes all running with 512Kbit/s network speed in the unmodified version and in the modified version running the last three nodes with 1GBit/s network speed instead of 512Kbit/s. The regression coefficient between real measured time and calculated with our model is $r_{unmodified}$ = 0.985 for the unmodified algorithms, and respectively $r_{modified}$ = 0.983 for the modified algorithms. That means the real measured time values are very close to our model and the behavior of the real measured values to those of the model are identical. Please note that the values with a smaller amount of data have more deviation than the values for a higher amount of data. The reason is that in a Service-Oriented Architecture (SOA) latencies can occur and therefore for a lower amount of data the model is not accurate enough.

IV. Conclusion and Further Work

In this work we presented a multi-layered blackboard approach for database query optimization in heterogeneous environments that eases the way of handling not only query-related optimization problems.

Furthermore it covers data- and resource-specific aspects as well, by assigning costs and rating decisions made in the query optimization process. The feasibility of such an approach was shown by an exemplifying evaluation model containing a small usage scenario. Because of the modularity of the proposed blackboard approach it can be very easily extended by incorporating sophisticated operational algorithms that benefit from large-scale rang of resources provided by Clouds. A directions for further research is specifically to cope with scalability for the query optimization process. We envision various ways, ranging from optimization of fragmented QET to a virtualized distributed blackboard.

References

[1] D. Bitton, H. Boral, D. DeWitt, and W. Wilkinson, "Parallel algorithms for the execution of relational database operations," *ACM Transactions on Database Systems*, vol. 8, no. 3, pp. 324–353, Sep 1983.

[2] D. Corkill, "Blackboard Systems," *AI Expert*, vol. 6, no. 9, January 1991. [Online]. Available: http://mas.cs.umass.edu/paper/218

[3] W. Mach and E. Schikuta, "Parallel algorithms for the execution of relational database operations revisited on grids," *International Journal of High Performance Computing Applications*, vol. 23, no. 2, pp. 152 – 170, Summer 2009.

[4] P. P. Beran, G. Habel, and E. Schikuta, "SODA - A Distributed Data Management Framework for the Internet of Services," *Grid and Cooperative Computing, International Conference on*, vol. 0, pp. 292–300, 2008.

[5] A. Kemper and G. Moerkotte, "Query optimization in object bases: Exploiting relational techniques," in *Query Optimization in Next Generation Database Systems*, J.-C. Freytag, D. Maier, and G. Vossen, Eds. Morgan-Kaufmann, 1993, pp. 63–98.

[6] W. Mach and E. Schikuta, "Performance analysis of parallel database sort operations in a heterogenous grid environment." in *CLUSTER*. IEEE, 2007, pp. 525–533. [Online]. Available: http://dblp.uni-trier.de/db/conf/cluster/cluster2007.html#MachS07