Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 156–165

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

A Multi-Staged Blackboard Query Optimization Framework for World-Spanning Distributed Database Resources

Peter Paul Beran[1], Werner Mach, Erich Schikuta, Ralph Vigne

University of Vienna, Workflow Systems and Technology Group, Rathausstr. 19/9, A-1010 Vienna, Austria

Abstract

With the advent of distributed computing, particularly since the emergence of Grids, Clouds and other Service Oriented Computing paradigms, the querying of huge datasets of distributed databases or data repositories on a global scale has become a challenging research question. Currently, beside various other topics, two major concerns in this research area have to be addressed: data access & integration and query execution planning. Our research effort addresses the second issue, namely the query optimization of distributed database queries. Hereby we consider a variety of different heterogeneous and homogeneous infrastructures, parallel algorithms, and huge datasets, which span across several virtual organizations (VOs) with usually no centralized authority. This paper introduces a novel heuristic framework for the optimization of query execution plans (QEP) on a world-wide scale. Our work is based on a multi-staged blackboard mechanism to determine which available data, resources and operations have to be considered to perform a query optimally. Moreover, an evaluation scenario proves our findings that even small changes in the selection of e.g. sort operations for a query execution tree (QET) lead to significant performance improvements.

Keywords: Distributed Databases, Query Optimization, Blackboard Method, Service Oriented Infrastructures.

1. Introduction

Growing database demands, new technological developments and new computing paradigms, as Grid and Cloud computing, unleashed new developments in the database technology sector. Meanwhile a steadily increasing proliferation of more and more inexpensive resources, such as processor, memory and hard disk, led to significant developments of sophisticated parallel database systems in the last decades [1]. These databases either operate on row-oriented or column-oriented schemes. Representatives of row-oriented databases are OracleDB, MySQL and PostgreSQL. Usually they provide a good performance for a generic set of use cases, but do not specifically focus on high performance. In contrast, column-oriented databases such as Vertica, Mariposa [2] or VoltDB establish mechanisms to provide high performance, but are mostly bound to a very particular execution environment. In generally parallelism in both types is achieved by facilitating intra- and inter-operator parallelism [3] and making use of computational resources (CPUs) to support a multitude of incoming query requests in parallel.

Email addresses: peter.beran@univie.ac.at (Peter Paul Beran), werner.mach@univie.ac.at (Werner Mach), erich.schikuta@univie.ac.at (Erich Schikuta), ralph.vigne@univie.ac.at (Ralph Vigne)

[1]Corresponding author (Phone: +43 (1) 4277-39510, Fax: +43 (1) 4277-9395)

doi:10.1016/j.procs.2011.04.017

In a loosely coupled environment, where the number and kind of participating databases is unstable, and the databases origin from different vendors, the mechanisms to perform a distributed query optimization and execution are still being investigated and improved. In such environments the driving factors of query optimization are cost (e.g. data shipment costs) and time (e.g. execution time). A further problem arising is that data sets are physically spread all over the world distributed onto many independent organizational domains. This leads naturally to the problem that in such environments the optimization of QEPs is an even more challenging task due to the heterogeneous nature of the infrastructure and the restricted and sparse information on the specific characteristics (functional or non-functional) of the involved query execution components.

Therefore we propose a solution approach that makes use of a blackboard mechanism [4] originating from the artificial intelligence community. This mechanism allows to solve the NP-hard task of distributed query optimization [5] by applying cost-based heuristics using an A^*-algorithm. It allows to find a (near) optimal solution by assigning costs to each decision taken in the optimization process. Finally, the best sequence of decisions obtains the lowest cost value and is therefore treated to be optimal. The specific characteristic of the optimization framework presented in this paper is the extension of the blackboard method to a hierarchical, multi-staged approach. This allows to cope with heterogeneous environments as well as unbalanced information on the infrastructure. Thus our framework allows to perform database queries on distributed databases located on different sites that are possibly managed by disjunctive organizations. Applications all over the world could benefit from our framework such as experiments in the area of high-energy physics experiments (e.g. ATLAS [6]) or in the biotechnology sector (e.g. Biobanks [7]), where a huge amount of data is divided among a multitude of partners.

This paper is organized as follows. In Section 2 a short overview of related work is provided. Section 3 presents our novel multi-staged blackboard architecture that consists of four stages covering different aspects of the query optimization in heterogeneous environments. In Section 4 we provide an exemplary use case scenario to illustrate the query optimization abilities of the multi-staged blackboard based on the TPC-H benchmark database schema. In Section 5 a practical evaluation and analysis is outlined to prove our observations that even small modifications in the structure, sequence and choice of operations in a QET can lead to huge benefits in the query optimization approach. The paper closes with a look at future developments and research directions in Section 6.

2. Related Work

In Nolle et al. [8] the authors present DARBS, a Distributed Algorithmic and Rule-based blackboard System for the automatic interpretation of nondestructive evaluation (NDE) data and the control of plasma deposition processes. Similar to our approach the blackboard is used as a central point of information that is updated and queried by generic agents. However, in our approach we distinguish between agents (experts) of different problem domains, allowing us to apply them more accurately in our optimization process.

Concerning the query optimization challenge a lot of work has been published focusing on different aspects of the whole problem [9, 10]. In recent work of Hameurlain and Morvan [11] standard optimization techniques for database queries are presented, also arguing about query optimization in large scale environments that can be accomplished using a broker-based approach or an approach based on mobile agents. Again these agents can be compared to the experts of a blackboard system that provide specific knowledge to the blackboard to solve problems or parts of problems of the overall optimization task.

3. Multi-Staged Blackboard Query Optimization

A blackboard system is a technique from artificial intelligence suitable especially for complex problems with an uncertain or incomplete knowledge. The blackboard method allows to solve NP-hard problems heuristically using an A^*-algorithm. This algorithm offers possibilities to prune the search space and thus avoid exhaustive searching. It concentrates only on promising solutions instead of taking every possible solution into account. It is based on the aggregation of knowledge in terms of collecting ideas (rules) covering a specific problem domain from different regions, each region representing an expert. Moreover by iterative re-formulation and alteration it is possible to deduce new rules out of existing ones. This is done until an appropriate solution for the given problem has been found, otherwise it can be assumed that the required knowledge and thus the expert is missing. The traditional notion of a blackboard was originally introduced by [12] and consists of the following components:

1. A **global blackboard** representing shared information space considering input data and partial solutions.
2. A **knowledge base** consisting of many independent regions, each owned by a single expert and containing its knowledge. Communication between different regions is accomplished via the global blackboard.
3. A **control component** that dictates the course of activities (phases) for the problem solving approach. In order to make reasonable decisions each region has to provide cost estimations for all applied operations to generate a cost-based decision tree.

In this paper we extend the blackboard method by a hierarchical, multi-stage approach. The typical situation of the infrastructure we face is a heterogeneous landscape with incomplete information of the possible components of the query plan. This is due the fact that there is no centralized control and the components taking part on the query execution are located in different administrative domains. Naturally the information on technical characteristics of components of these domains from outside is limited due to political, economic or technical reasons and restricted to a defined interface. Such domains can comprise both single nodes, as database servers, and large organizational units, e.g. Clouds. We call such domains Virtual Organizations (VOs), where resources may be dispersed geographically but function as a coherent unit.

The concept of Virtual Organizations (VO) was born from Grid Computing and is described as flexible, secure, coordinated resource sharing among dynamic collections of individuals, institutions, and resources [13]. In the context of database query execution a VO is a temporary coalition of geographically dispersed organizations to collaborate and share their resources in order to fulfill the query requests. A virtual organization is typically represented by an organizational agent, which is the single (or main) point of entry for communication with the outer world (external agents). The motivation for establishing a VO is manifold and can be based on technical, economical, political, administrative, organizational, etc. reasons. So for example, on the one hand some administrative VOs restrict the access to data sets in question to a defined interface hindering the free access to the data. On the other hand technically motivated VOs are defined on affinity of the specific nature of the underlaying infrastructure, e.g. storage techniques, computational resources, network layers or algorithmic issues.

Distributed QEPs span typically across several VOs in practice. The optimization of such plans is an even more tedious task. For a classical, centralized optimization process it is typically not possible "to see beyond the border" of a VO. Therefore specific properties of the *local* resources of a VO can not be used for the optimization of the QEP and the *global* optimization on top of the VOs is limited. This motivated us to develop a method which shifts the local optimization process to the VOs and integrates the local optimized QEP into a global optimized QEP on top of the VOs. Due to the situation that a VO itself is built of several other VOs, we developed a hierarchical, multi-stage optimization process, which allows to cope with nested VOs too. In this endeavor we define a generalized hierarchical, recursive backboard approach for calculation of the optimized QEP using two (simple) lists: An *OpenList*, which stores known (generated) but not visited nodes, and a *CloseList*, which stores all already visited (expanded) nodes. An expansion of a node is defined by the generation of all successors of this node i.e. the OpenList contains all generated, but not expanded nodes, therefore the CloseList contains all expanded nodes.

The interpretation of this algorithm in the context of QEPs and resources is the following: A situation in the specific case defines a set of resources describing a (partial) QEP. A node in the graph denotes a (partial) QEP. An edge in the graph from node v to node w denotes a (under many) possible extension of the QEP v by a new resource. The function *best(OpenList)* returns the node v from OpenList with costs $v.c + h(v)$ is minimal. Algorithm 1 shows how VOs are incorporated during the calculation of the QEP. With reference to the blackboard method, it is assumed that when the QEP is presented on the blackboard, different agents offer solutions for different VOs of the workflow. During expansion of a node (Algorithm 2) it is differentiated between a "public" expansion, where the costs can be calculated on the top level, and a "local" expansion, where the costs are calculated hidden within a VO. A local expansion leads normally to an optimization of a partial QEP within of the VO. The indirect recursion in the algorithms allows for nested VOs too.

In the following we assume a VO landscape motivated by technical problem domains, which is divided into four stages (Fig. 1). Each of these stages consists of a blackboard targeting a specific problem domain. In a step-wise approach the optimization takes place either within a stage (local iteration) or spanning all stages (global iteration). Whenever a new solution path emerges, due to changing conditions, the blackboard reacts dynamically to these changing conditions.

```
Input: Start node S of the initial QEP
Output: Node describing optimized QEP (if available)
$OpenList = [S]$; $BlockedList = []$;
while $OpenList \neq []$ do
    $Act$ = best($OpenList$);
    if $Act == Goal$ then
        return FOUND;
    end
    $OpenList = OpenList \setminus [Act]$;
    foreach node v1 in expand(Act) do
        if $v1 \notin OpenList \wedge v1 \notin BlockedList$ then
            $v1.c = Act.c + h(Act)$;
            $OpenList = [v1] + OpenList$;
        else
            if $Act.c + h(Act) < v1.c$ then
                $v1.c = Act.c + h(Act)$;
            end
        end
    end
end
// Goal not reached, i.e.  no QEP possible
return FAIL;
```

Algorithm 1: `Optimize` - The Extended Blackboard Algorithm for Hierarchical QEP Optimization

```
Input: Node Act to be expanded
Output: List of expanded nodes of Act (if available)
$L = []$;
foreach node $A.x \in$ expansion of (Act) do
    if $x \notin VOs$ then
        add $Act.x$ to $L$;
    else
        add call_VO_agent.optimize($Act.x$) to $L$;
    end
end
return $L$;
```

Algorithm 2: `Expand` - Expansion of a Node

Generally two different types of parameters can be identified. Additionally we introduce the concept of priorities that allow to weight parameters according to their importance, 0 for optional and 1 for mandatory parameters.

- **Continuous parameters** denote parameters which are to be minimized or maximized by the cost function. The value domain is limited by a border value that denotes the least acceptable value. Values beyond the border result in infinite costs. The costs for each parameter choice are computed by assigning the weakest parameter the cost value 1 and scaling all other parameters according to the weakest one.

- **Guarantee parameters** describe parameters that have to meet a specific value. These parameter choices are either evaluated to “true” or “false”. Parameters conforming to the required value result in costs of 0, all other parameters result in infinite costs.

To allow for cost-based decisions within each blackboard stage, the different choices and their attached parameters have to be comparable. Therefore the authors propose a common language vocabulary that allows to map parameters to a specific concept within an ontology, as presented in [5]. Moreover, the different parameter values must be of the same value-type and have to be compliant to a nominal, ordinal, interval or ratio scale [14]. If all conditions are satisfied the heuristic algorithm is able to choose the best alternative (node) for the next step to reach the overall optimization goal.

Sometimes the heuristic approach is stuck in a local optimum and will not find the global optimum. This is clear due to the nature of the algorithm, but can be neglected as a rather good solution (local optimum) found very fast is often better than the best solution (global optimum) that needs lot of time for computation. Due to the NP-hard characteristic of the optimization problem a *best* solution to find is not feasible. A possible solution for this problem applies a hybrid strategy that performs ballooning. Thus, in a first step the blackboard looks for the most trivial solution to the optimization problem, without considering the costs of parameters (decisions). Then it triggers the start of the constructed QEP. Meanwhile the blackboard continues its operation and tries to find the optimal solution. When a solution is found the blackboard stops the ongoing query execution process and replaces execution paths that are not optimal and do not have started yet. After replacement it continues the query execution, which now follows the optimal strategy.

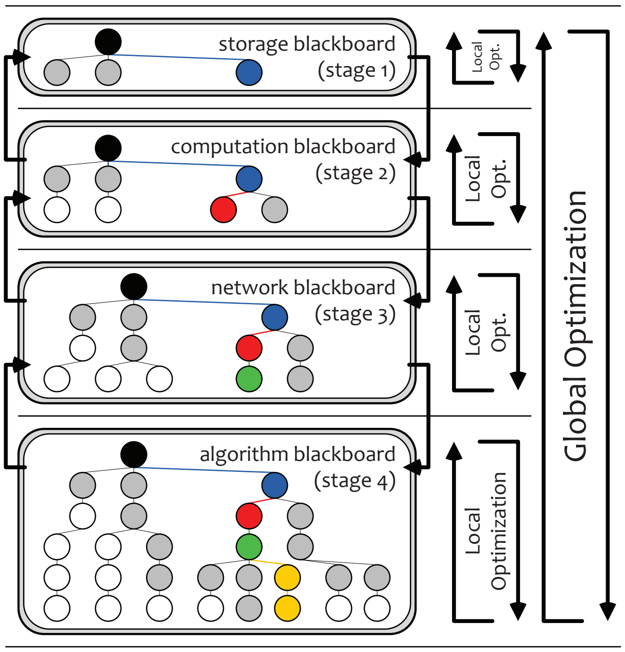


Figure 1: Multi-staged Blackboard Architecture for Distributed QEP

The multi-staged blackboard consists of four stages, each being a blackboard on its own:

1. **Storage Blackboard:** It covers 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. Referring to our preceding research [15] 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 cause the transmission time for input and output data to change dramatically.
2. **Computation Blackboard:** It focuses on the assignment and allocation of computing resources and benefits from the optimization capabilities for homogeneous and heterogeneous environments, considering time-invariant issues (e.g. CPU frequency) and time-variant issues (e.g. actual CPU load). Using our SODA [16] framework, a SOA-enabled prototype of a distributed database system, we successfully demonstrated how such a resource classification and characterization can be applied to heterogeneous distributed resources.
3. **Network Blackboard:** It requires the knowledge of the previously chosen storage and computational resources to determine the network connectivity in terms of time-invariant issues (e.g. network link bandwidth) and time-variant issues (e.g. actual network bandwidth) between these resources.
4. **Algorithm Blackboard:** It covers the query optimization itself and consists of many consecutive regions, each containing a sequence of expansion rules to transform an QET. At the beginning a user query is translated into a normal form similar to a most costly normal form [17] 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 classical rules of Ceri et al. [18] for distributed databases and were adapted for the use in parallel and distributed data repositories. Referring to our previous work [19] they are grouped according to the database operation types: 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 based on the given ones. However all transformation rules only affect the syntactic notation of the QET, which in turn changes the execution sequence of the query, but does not affect the query result.

4. Use Case Scenario of Blackboard-based Optimization Process

The following scenario details and clarifies our approach and defines how to measure the enhancements of our multi-staged blackboard. The chosen database model was derived from the TPC-H benchmark. For the scenario, the PART and PARTSUPP tables were considered to perform a query that uses two projections (π), two selections (σ), a join ($\bowtie$) and a sort ($\downarrow\uparrow$) operation, as depicted in the following SQL expression:

```
SELECT p.*, ps.ps_availqty, ps.ps_supplycost
FROM part p INNER JOIN partsupp ps ON p.p_partkey = ps.ps_partkey
WHERE ps.ps_availqty >= 100 AND ps.ps_supplycost <= 1.99
ORDER BY p.p_size
```

To perform this operation in an "optimal" way the challenge is to take all storage-, computation-, network- and algorithm-related issues into account using a cost-based optimization model. Moreover, using the previously defined priority concept, we can incorporate request related aspects. Considering data-intensive queries the priority factor for storage-related parameters will be high. On the contrary, having compute-intensive queries will lead to a high priority regarding all parameters in the computation stage. In our scenario we assume that all parameters are "mandatory" and equally important, thus having the priority value 1 applied.

4.1. Evaluation Environment

In our approach we use SODA (Service Oriented Database Architecture) as presented in [16]. It consists of small building blocks – implemented as Web services – which can be plugged together almost in a LEGO$^{\text{TM}}$ like manner to shape a QET. It provides the business logic for a distributed query execution, specifically supporting autonomous databases in heterogeneous environments, and acts as a distributed query engine. Each Web service implements an atomic database operation such as projection, selection, join or sort. Hence it benefits from novel operation algorithms that can be incorporated into the environment.

4.2. Query Optimization

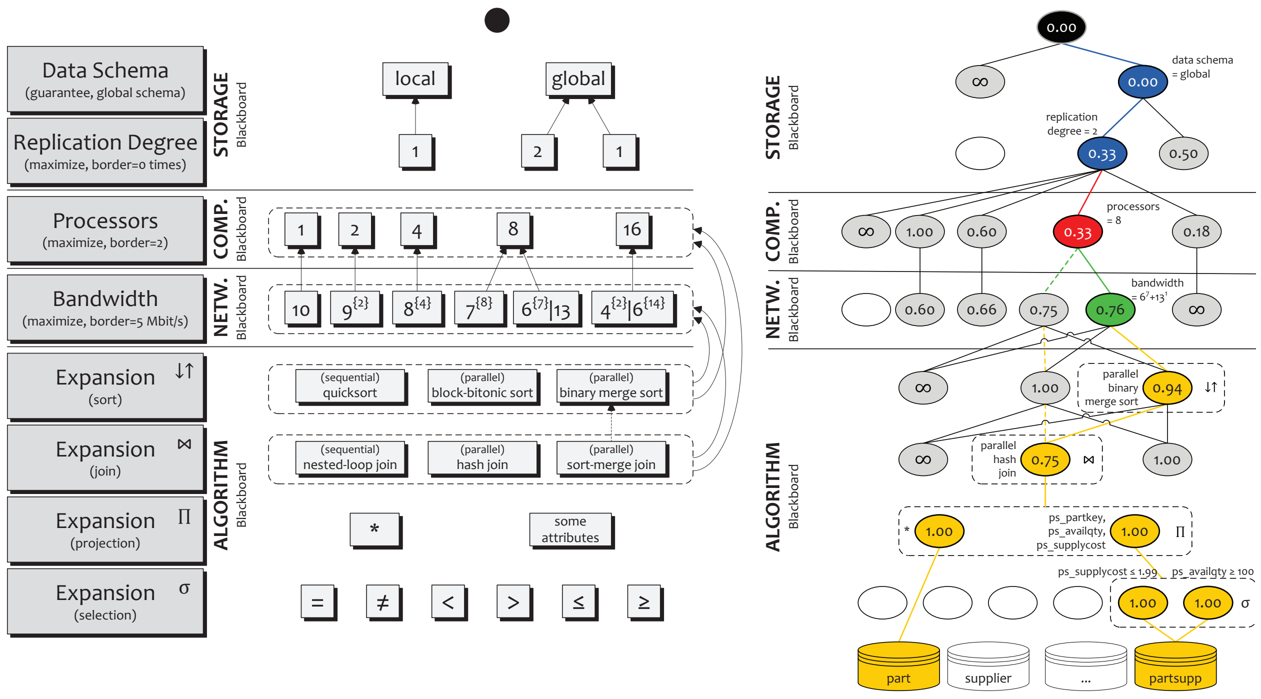
The initial blackboard is given in Fig. 2(a) where each possible parameter value is depicted by a node (rectangle) in the corresponding layer. Edges between nodes indicate logical relations in terms of dependencies, e.g. if data sources with a global schema are used then the replication degree can be either 1 or 2. Between different layers no edges are drawn because 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 indicates how many nodes achieve the bandwidth.

Storage Blackboard. Concerning data-related issues, the *Data Schema* and the *Replication Degree* parameter costs (Equ. 1) have to be optimized. As the provision of a global schema is compulsory, the right ("global") node is chosen (costs: 0.0). All nodes providing only a local schema are assigned with infinite costs and are not further considered. Due to the fact that we do not need replicated data, the replication degree border value is set to 0. Nevertheless the node with replication degree 2 (costs: 0.33) is selected instead of the node with replication degree 1 (costs: 0.50), because if one data location becomes unavailable we can easily switch to another replica.

$$c_{storage}(x) \quad = \quad (1 - \left\{ \begin{array}{l} border_{data\ schema} = x_{data\ schema} \ldots 1 \\ border_{data\ schema} \neq x_{data\ schema} \ldots 0 \end{array} \right\}) * \infty * 1.0 + \frac{border_{replication\ degree} + 1}{x_{replication\ degree} + 1} * 1.0 \qquad (1)$$

Computation Blackboard. Concerning computation-related issues, the number of available *Processors* has to be optimized (Equ. 2). For the computational resources a lower border of at least two processors is mandatory. If only one processor is available the costs are infinite. In the first attempt the right-most node (costs: 0.18) providing 16 cores is selected.

$$c_{computation}(x) \quad = \quad \frac{border_{processors} + 1}{x_{processors} + 1} * 1.0 \qquad (2)$$



(a) Initial Blackboard with possible Choices (b) Final Blackboard with optimal Solution

Network Blackboard. Concerning network-related issues, the network *Bandwidth* has to be optimized (Equ. 3). Therefore the bandwidth of each processor must not under-run 5Mbit/s. If the bandwidth of at least one processor drops below 5Mbit/s the costs are infinite. Already the next node – under the 16 processors node – does not fulfill the bandwidth requirements, thus the costs become infinite. Due to this fact a former decision has to be revised switching to the configuration with only 8 cores (costs: 0.33), as shown in 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 possible choice is, at the moment, not further investigated (costs: 0.76).

$$c_{network}(x) \quad = \quad \frac{border_{bandwidth} + 1}{\frac{\sum_{i=1}^{processors}(x^i_{bandwidth})}{processors} + 1} * 1.0 \tag{3}$$

Algorithm Blackboard. Concerning operation-related issues, only basic cost functions for relational operations have to be considered. First of all the SQL-query is translated into its corresponding normal form. Based on this normal form the QET can be created consisting of two selections (σ), two projections (π), a join ($\bowtie$) and a sort ($\downarrow\uparrow$) layer (Equ. 4). Then the optimizer applies the required operators. Among them the three different sort operators ($\downarrow\uparrow_q$ for quicksort, $\downarrow\uparrow_{pbb}$ for parallel block-bitonic sort, $\downarrow\uparrow_{pbm}$ for parallel binary merge sort) exist. For the expansion of the projection (costs: 1.00) there is only one choice for $*$ (return all attributes) and the projection of three attributes (`ps_partkey` needed for join, `ps_availqty` and `ps_supplycost`). Thus the corresponding nodes are selected. The same applies to the two selection operations, a greater-equal operation for the attribute `ps_supplycost` and a less-equal operation for the `ps_availqty` attribute. Both nodes are chosen in the final QET. For the join operation three different algorithms can be chosen, a sequential nested-loop join, a parallel hash join or a parallel sort-merge join. We skip the nested-loop join because 8 processors allow to perform a parallel algorithm. Furthermore we decide to use the hash join algorithm, because it is more flexible and does not need to have the tuples sorted by the join attribute (`partkey`). Moreover we require to sort the joined tuples by the `p_size` attribute, thus we need to apply one of the sort algorithms. In a first attempt the $\downarrow\uparrow_{pbb}$ (costs: 0.99) is chosen, instead of the $\downarrow\uparrow_{pbm}$ (costs: 1.00) 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 the costs can be reduced to 0.94. Therefore a new solution covering the alternative bandwidth configuration and the $\downarrow\uparrow_{pbm}$ is now 0.05 cheaper than the former one using the $\downarrow\uparrow_{pbb}$. Thus the chosen node in the network stage has to be changed to benefit for a modified parallel binary merge sort. Regarding the two branches that contain on the one side only a single projection (∗) and on the other side a sequence of two selections (≤, ≥) and a projection (`ps_partkey`, `ps_availqty`, `ps_supplycost`) a maximum function can be applied, because these two sequences are performed on different databases and therefore can run in parallel.

$$c_{algorithm}(x) = max(\pi^{*}_{costs}\ ,\ (\sigma^{\leq}_{costs} + \sigma^{\geq}_{costs} + \pi^{pk,aq,sc}_{costs})) + \bowtie^{hash}_{costs} + \downarrow\uparrow^{pbm}_{costs} \tag{4}$$

$$\begin{array}{ccccccccc} c_{overall}(x) & = & c_{storage}(x) & + & c_{computation}(x) & + & c_{network}(x) & + & c_{algorithm}(x) \\ 5.11 & = & 0.00 + 0.33 & + & 0.33 & + & 0.76 & + & 2.00 + 0.75 + 0.94 \end{array} \tag{5}$$

Finally the estimated costs for the best solution (Fig. 2(b)) are 5.11, according to Equ. 5. The solid bold line indicates the optimal path of decisions taken to perform the query in an optimal way.

5. Practical Evaluation and Analysis

To prove our statements of the preceding sort model, where we reconsider a decision because of an alternative network configuration, we implemented the discussed sort algorithms using SODA. The performance analysis of our algorithms is based on a derived version of the TPC-H benchmark, only adopting the `PART` table. Therefore we prove our last decision that leads us to use the modified parallel binary merge sort instead of the parallel block-bitonic sort was justifiable. Using the *DBGEN* tool allows us to populate the database. 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), each running an Ubuntu Linux 8.10, was used. For the measurements the workflow (Fig. 2) containing the parallel binary merge sort was executed in an unmodified and a modified style. In case of the modified version the network speed of the last three nodes (#2,#3,#4) was fixed to 1GBit/s. For all other nodes the network connection speed has been throttled between 512Kbit/s and 1GBit/s during the runs to simulate a typical heterogeneous environment and proof our hypothesis.

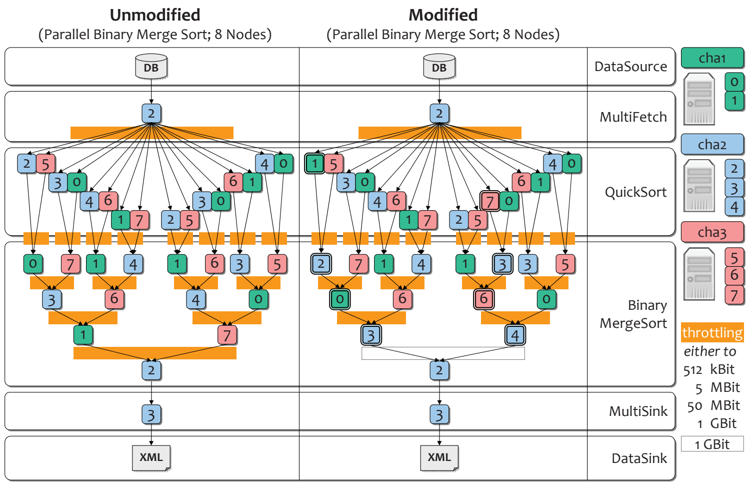


Figure 2: Parallel Binary Mergesort Workflow Configuration

5.1. Modified Sort 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 than the binary merge sort, as outlined 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 (Equ. 6), because here only one processor is necessary doing the merge for $\frac{n}{2}$ costs.

$$\underbrace{\frac{n}{2p}log(\frac{n}{2p})}_{suboptimal} \quad \underbrace{+\frac{n}{2p}}_{optimal} \quad \underbrace{+log(p)-1}_{postoptimal_I} \quad \underbrace{+\frac{n}{2}}_{postoptimal_{II}} \tag{6}$$

If we choose for this “bottleneck” the nodes of the heterogeneous environment 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. It is intuitive to use nodes with the highest network performance in the $postoptimal_{II}$ phase as laid out in the 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. A speedup and scale-up analysis [16] for the used SODA environment also verifies this issue. Please note that we used a heterogeneous environment for our performance analysis and not a homogeneous environment like common sort benchmarks (e.g. Terasort), that means a comparison with these benchmarks does not make sense.

5.2. Environment Configuration for Sort Workflow

According to our presented orchestration algorithm we place the postoptimal phase of the binary merge sort algorithm on the node with the highest performance and best bandwidth connection. The practical results justify 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. Unfortunately due to the Java and XML nature of the SODA prototype we were not able to run the sort workflow with more then about 2 MB.

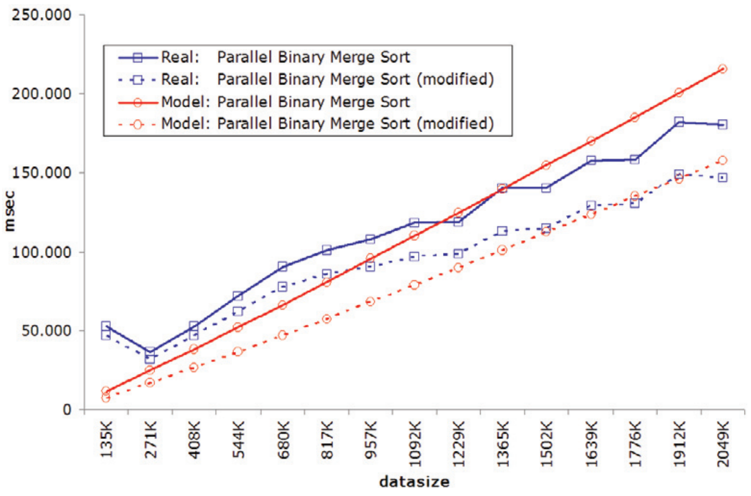


Figure 3: Real and Model Performance Behavior of Parallel Binary Merge Sort (unmodified and modified)

The performance analysis of the analytical model shows that a smart orchestration of the available nodes with heterogeneous performance characteristics results in an increased performance behavior, as depicted in Fig. 3. The regression coefficient between real measured time and calculated with our model is 0.985 for the unmodified algorithms, and respectively 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 smaller amount of data the model is not accurate enough.

6. Conclusion and Further Work

In this work we presented a novel hierarchical, multi-staged blackboard framework for database query optimization in heterogeneous environments that eases the way of handling not only query-related optimization problems, but also covers storage-, computation- and network-related aspects. This can be accomplished by assigning costs to decisions made in the overall query optimization process and by weighting the importance of these decisions using priorities. The feasibility of such an approach was shown by providing an exemplified evaluation model containing a small use case scenario. Because of the modularity of the proposed blackboard it can be extended very easily by incorporating sophisticated operational algorithms that benefit from large-scale rank of resources provided by Clouds (e.g. Amazons EC2/S3). Moreover, we demonstrated in an evaluation section that even small changes in the QET result in considerable benefits in the overall query execution approach. In the future we plan to prove our hypothesis in how we arrange and order the different stages (storage, computation, network, algorithm) of the blackboard. Moreover by doing some simulations we aim to determine the relative importance of parameters per stage allowing us to perform decisions on crucial parameters first before less important parameters have to be considered for the optimization.

References

[1] D. Bitton, H. Boral, D. J. DeWitt, W. K. Wilkinson, Parallel algorithms for the execution of relational database operations, ACM Transactions on Database Systems 8 (3) (1983) 324–353.

[2] M. Stonebraker, P. M. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, A. Yu, Mariposa: A wide-area distributed Database System, The VLDB Journal 5 (1996) 048–063.

[3] D. DeWitt, J. Gray, Parallel Database Systems: The Future of High Performance Database Systems, Communications of the ACM 35 (6) (1992) 85–98.

[4] H. Wanek, E. Schikuta, I. Ul-Haq, Grid Workflow Optimization Regarding Dynamically Changing Resources and Conditions, Grid and Cloud Computing, International Conference on 0 (2007) 757–763.

[5] E. Vinek, P. P. Beran, E. Schikuta, Mapping Distributed Heterogeneous Systems to a Common Language by Applying Ontologies, in: PDCN 2011: Proceedings of the 10th IASTED International Conference on Parallel and Distributed Computing and Networks, IASTED/ACTA Press, 2011, pp. 190–197, accepted for publication.

[6] The ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, in: JINST 3 S08003, 2008, p. S08003.

[7] J. Eder, C. Dabringer, M. Schicho, K. Stark, Data Management for Federated Biobanks, in: S. Bhowmick, J. Kueng, R. Wagner (Eds.), Database and Expert Systems Applications, Vol. 5690 of Lecture Notes in Comp. Science, Springer Berlin / Heidelberg, 2009, pp. 184–195.

[8] L. Nolle, K. Wong, A. Hopgood, DARBS: A Distributed Blackboard System, in: 21st SGES International Conference on Knowledge Based Systems and Applied Artificial Intelligence, 2001, pp. 162–169.

[9] U. Srivastava, K. Munagala, J. Widom, R. Motwani, Query Optimization over Web Services, in: Proceedings of the 32nd international conference on Very large data bases, VLDB '06, VLDB Endowment, 2006, pp. 355–366.

[10] D. Braga, S. Ceri, M. Grossniklaus, Join Methods and Query Optimization, in: SeCO Workshop, 2009, pp. 188–210.

[11] A. Hameurlain, Evolution of query optimization methods: From centralized database systems to data grid systems, in: S. Bhowmick, J. Kueng, R. Wagner (Eds.), Database and Expert Systems Applications, Vol. 5690 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2009, pp. 460–470.

[12] D. Corkill, Blackboard Systems, AI Expert 6 (9).

[13] I. Foster, C. Kesselman, S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International Journal Supercomputer Applications 15 (3).

[14] S. Chaari, Y. Badr, F. Biennier, C. BenAmar, J. Favrel, Framework for Web Service Selection Based on Non-Functional Properties, International Journal of Web Services Practices 3 (2) (2008) 94–109.

[15] W. Mach, E. Schikuta, Parallel algorithms for the execution of relational database operations revisited on grids, International Journal of High Performance Computing Applications 23 (2) (2009) 152–170.

[16] P. P. Beran, G. Habel, E. Schikuta, SODA - A Distributed Data Management Framework for the Internet of Services, in: GCC '08: Proceedings of the 2008 Seventh International Conference on Grid and Cooperative Computing, IEEE Computer Society, Washington, DC, USA, 2008, pp. 292–300.

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

[18] S. Ceri, G. Pelagatti, Distributed Databases, Principles and Systems, McGraw-Hill Book Company, New York, NY, USA, 1985.

[19] E. Schikuta, P. Kirkovits, A Knowledge Base for the Optimization of Parallel Query Execution Plans, in: ISPAN '97: Proceedings of the 1997 International Symposium on Parallel Architectures, Algorithms and Networks, IEEE Computer Society, Washington, DC, USA, 1997, p. 331.