An Overview of Parallel Query Optimization in Relational Systems

Abdelkader Hameurlain, Franck Morvan
Institut de Recherche en Informatique de Toulouse IRIT, Université Paul Sabatier
118, Route de Narbonne, 31062 Toulouse cedex, France
E-mail: {hameur, morvan}@irit.fr

Abstract

In this paper we try to describe synthetically the main problems which arise in the design of parallel relational database systems. Due to the breadth and the depth of the topic, it is obviously very hard to capture all good issues in a short paper. The issues that are tackled include data placement, static and dynamic query optimization, resource allocation, statistics extraction, and cost estimation.

1. Introduction

The optimizer's role is to generate, in a considered search space and for a given SQL query, an execution plan close to optimum (or optimum). The optimization goal is to minimize response time, maximize throughput while simultaneously minimizing optimization costs. The general problem for query optimization may be expressed as follow [17]: given a query q, a space of execution plans E, and a cost function cost(q) associated with an execution plan p ∈ E, find the execution plan calculating q such as the cost(q) is minimal. Generally, the design of an optimizer can be split in three components [17]: <*Search Space* which defines the syntax of all aspects relating to execution, *Search Strategy which* generates a close to optimum execution plan, *Cost Model* which assigns a cost to an execution plan>.

Due the importance and complexity of the optimization problem in relational queries and since publication of [59], the database community has expended much effort to develop optimization methods in different way for various data base systems: centralized (uniprocessor), distributed, deductive, and parallel. A complete, and elegant review of optimizers and their methods in relational systems is described in [12]. Those optimizers are based on query rewrite and dynamic programming approach relies on the partial optimum principle.

In parallel database systems, and parallel query optimization, [15, 28, 41, 61] provide an excellent review: the authors identify issues that are relevant, discuss several open issues, and propose some directions of research offering new challenges.

Parallel (relational) query optimization methods can be considered as an extension of conventional relational query optimization methods which integrate the parallelism dimension. Indeed, the generation of a parallel execution plan, close to optimum, is either a two-phase approach [2, 4, 5, 13, 19, 20, 25, 26, 27, 30, 31, 39] or one-phase approach [9, 10, 14, 43, 44, 50, 56, 58, 64]. The two-phase approach comprises two distinct stages in sequence: (i) execution plan generation (i.e. logical optimization followed by physical optimization), and (ii) resource allocation to the execution plan (i.e. parallelism extraction followed by physical allocation of resources). As for the one-phase approach, the plan generation and resource allocation are packed into one integrated component [49].

Whatever approach is used to generate a parallel execution plan, four different types of optimization methods exist: static, dynamic, hybrid, and quasi-hybrid. This last method [3, 34, 39] first applies a static method, followed by a dynamic method which may re-invoke, contrary to the purely hybrid method [22], the query optimizer (e.g. physical optimization) during execution for corrective purposes.

The rest of the paper is structured as follows: the Section 2 specifies elements important for the relevant choice of initial data placement and describes the data skew phenomenon. Following the presentation of the main characteristics of parallel query optimization methods, in Sections 4, and 5, we present a few parallelization-optimization methods, static and dynamic, separating the one-phase and two-phase approaches. Section 6 hints to the importance and complexity of obtaining accurate and efficient cost estimates. Finally, Section 7 gives a glance at problems which constitute a challenge for future parallel relational database systems.

0-7695-0680-1/00 $10.00 © 2000 IEEE

2. Data partitioning and data skew

The main problem concerning data partitioning, also called data placement, involves finding and keeping the best compromise between processing costs and the cost of data communication and control. More precisely, relatively accurate information concerning databases and applications is needed (e.g. SQL query characteristics, host variables) as well as information on the evolution of data and the extensibility of the parallel system. This allows the database administrator DBA to select meaningfully: (i) a data partitioning approach: full declustering vs partial declustering. [51] propose a detailed and complete study of data placement in shared-nothing systems, (ii) a partitioning degree for a relation, often called Home of relation, which corresponds to the number of nodes over which the relation is partitioned, and (iii) a data partitioning method as a function of available algorithms, of data access modes, and query characteristics (e.g. number of relations, number of predicates, query form).

Data skew phenomenon is mainly due to choices made during data partitioning approach (i.e. selection of partitioning degree for a relation), with placement method, and the quality of the quantitative informations the DBA has. Several types of skew have been identified [51, 62] tuple placement skew, selectivity skew, redistribution skew and join product skew. Data skew phenomenon generates, mainly, intra-operator load balancing problems which many considerably deteriorate the performance expected for parallel relational operators. Indeed, load balancing for intra-operator parallelism has been intensively studied [33, 42, 62]. Quite a few of the methods have been proposed as algorithms, mainly, for specific joins integrating appropriate mechanisms to avoid or solve different types of skew.

3. Main characteristics of parallel query optimization methods

Generally, a parallel query optimization method can be characterized by the following quartet and accompanying components < Context, Environment, Parallel Architecture Type, Search Space Nature>:

- *Context*: may be static, dynamic, hybrid, or quasi-hybrid.

- *Environment*: multi-user, or single-user. In this last case, the system's resources are allocated to the operations of a single query.

- *Parallel architecture type*: based on MIMD execution model, five parallel architecture types have been used as platforms to develop parallel DBMS. Shared-everything, shared disk, shared-nothing, hybrid architecture, and NUMA/COMA.

- *Search space nature*: this is mainly characterized by the considered or valid tree types (i.e. left deep trees, right deep trees, and bushy trees), and the different types and forms of parallelism considered.

The principal interest in this characterization is twofold: (i) to establish an optimization method classification to compare methods between themselves, and (ii) to describe the degree of complexity of methods with respect to their characteristic components.

In the following two Sections, we critically look at few query parallelization-optimization methods, first mostly static, then dynamic. Such critiques are relevant to highlight: (i) the guidelines for query parallelization strategies, (ii) the reasons why the proposed strategies become quite sophisticated in both static and dynamic contexts, and (iii) the central importance of parallel query optimization problems.

4. Parallelization strategies in static context

In this section, we describe scheduling strategies and (logical) allocation of resources taking into account parallelism form and space search type [13, 14, 17, 19, 20, 26, 27, 58, 64]. In the framework of two-phase parallel optimization methods, [13, 17, 26, 27] propose several scheduling strategies for a relational operator tree linked only by pipeline. After having observed the importance of communication overhead in the study NonStop SQL/PM DBMS and shown the interest of taking this into account, they develop a response time model leading to a trade-off between parallel execution and communication overhead. This is fundamental to achieve query response time improvements. Several approach scheduling, based on heuristics, are then proposed. They are inspired of the LPT (Largest Processing Time) heuristic and based on symmetrical and equilibrated algorithm classes [27]. These algorithms use pipeline and partitioned (intra-operation) parallelism but neglect precedence constraints which may exist depending on the operator tree.

As far as [19, 20] are concerned, they can be seen as a very elegant extension of the work proposed in [13, 17, 26, 27] which only consider the CPU resource in a single-user context. In this way, in [19, 20], optimizing and scheduling of execution plans on hierarchical parallel systems which communicate by message-passing are dealt with. Several heuristics are suggested which realize static simultaneous scheduling/mapping of operator fragments (a fragment is a chain of pipeline communicating operators) by taking into account multiple preemptable (e.g. CPU, disk, network) and non preemptable (e.g. memory) resource sharing between concurrent operators, the different types of parallelism (e.g. pipeline, independent and intra-operation), and data placement, in single- and multi-user contexts. However, the heuristics proposed, based on quite drastic assumptions, remain only valid in a static context.

In the one-phase approach, [58] propose a parallel program for processing of a query composed of N joins, for each form of search space (i.e. left deep tree, right deep tree, and bushy tree). The authors investigate in detail for each form of search space the memory size requirements, scheduling possibilities, and the ability to use the different types of parallelism. The study includes that case where resource memory is unlimited as well and the more realistic case where resource memory is limited. In the first case, the right deep tree is well suited to exploit parallelism at it's best. However, this structure is no longer the best when memory is limited. In this case, [58] propose to split the processing tree into many disjointed sub-trees such that all the relations of each sub-tree fit into memory. Temporary relations T1, T2,..., Tn will be stored on disks between two-phases[1]. The drawback of this approach is that the number of sub-trees increases with the number of base relations which do not fit into memory. Thus, this method reduces pipeline length and increases response time. Two elegant solutions have been proposed; the former is based on segmented right-deep trees [14], and the latter, on zig-zag trees [64].

5. Parallelization strategies in dynamic context

There is motivation to introduce a dynamic strategy of resource allocation [21, 22, 63]. This strategy is based on the will to use information not available at compile time and concerning mainly *resource availability* and *quasi-optimal* in metrics. The proposals [9, 10, 21, 22, 31, 39, 43, 44, 50] point out the importance not only load balancing [6, 8, 55, 56], but also correcting sub-optimal execution plans generated at compile time and detectable at run-time [3, 5, 34, 39]. The methods developed become very sophisticated and present in fact *'re-parallelization'* if not *re-optimization* strategies at run-time, far from simple resource allocation methods. The requirement for execution plan readjustment prescribes a new parallelization phase if not also a "new static optimization" (e.g. physical optimization). Furthermore, these recent strategies are not always purely dynamic, but can be also hybrid or *quasi-hybrid* [3, 5]. They are all based on the will to fully exploit the capabilities of the cost evaluator [18, 24].

Thus, [50] propose: (i) a RateMatch dynamic algorithm to determine the intra-operation parallelism degree for a single join operator which respects the initial data placement, and (ii) many alternative processor allocation methods to the operator instances by considering heuristics such as Random or Round-Robin, and criteria related to the system resources such as available memory, CPU or disk utilization. Both these algorithm classes can be combined to obtain a large variety of processor allocation strategies. The processor allocation algorithms proposed don't integrate the allocation constraints (i.e. data locality, data localization, pipeline and independent parallelism types). They separately integrate resources of the preemptable type and of the non preemptable one. The proposals of [56], which deepen the work proposed by [50], deal with the problem of dynamic load balancing of single simple-hash join queries in shared-nothing systems. The intra-query parallelism degree and the choice of the processors to process the sub-queries are determined in an "integrated" manner, and the current state of the system is considered with regard to the disk, memory and CPU resources. Several multi-resource load balancing dynamic strategies are proposed which enable to re-adjust at run-time the processor allocation on the basis of the operation I/O and CPU consumption. In this way, the parallelism degree adjustment in case of CPU-bottleneck is performed by systematically reducing the number of processors in keeping with the average CPU utilization [55]. As to the parallelism degree adjustment in case of memory-bottleneck, it is realized by systematically increasing the number of processors to limit memory contention, which may lead to the CPU-bottleneck symmetric problem. Moreover, the parallelism degree adjustment is performed from the optimal number of processors which is generally calculated from a more or less complex analytical expression (i.e. single-user or multi-user mode) associated to the most adequate join method. At completion of this adjustment, the problem is that keeping the same join algorithm may not be assured because a better algorithm is likely to exist for this new number of processors.

In XPRS adaptive scheduling approach [30], the method proposed is relies on maximum resource utilization (i.e. I/O bandwidth and processor capacities), by executing in parallel two fragments at their IO-CPU balance point. When one of the two fragments is completed, [31] propose a dynamic method to adjust the parallelism degree of this fragment depending on the partitioning method used (i.e. page partitioning or range partitioning). The implementation, on a shared-everything system, of this adjustment method is based on communication and cooperation between master and slave processors.

The most recent work of [9, 10, 43, 44] is particularly interested in multi-join processing, by considering the current state of the system in terms of multi-resource contention. [9, 43, 44] cover, more generally, complex decision query optimization in a shared-nothing system. The proposed optimizer determines dynamically the intra-operation parallelism degree for bushy tree join operators, and simultaneously allocates logically the resources. The authors suggest a four-level resource allocation heuristic (i.e. data locality, memory size, buckets, bushy tree serialization).

[1] A phase corresponds to a sub-tree execution.

Serialization [43] consists in introducing, in the operator tree, precedence constraints between some join operators unrelated by any data dependence. This models a new scheduling of operators in case of resource contention. This technique is also used if inter-operator load balancing is too difficult to achieve.

The sprite of recent work proposed by [39] is very close to the one of [9, 43], but they use a two-phase approach. The authors propose an algorithm, implemented in the Paradise database system [16] based on the static optimizer OPT++ [40], which detects sub-optimality of a complex query execution plan while the query is being executed, and attempt to correct the defects. It is based on statistics collected at certain key points during the execution of the query, and then used to optimize the query execution, either by improving resource allocation for this query, or by changing the execution plan concerning the part of the query not yet executed. Finally, in the context, more wide, of data integration systems, the proposals of [34] can be seen as an extension of the work proposed in [3, 5, 39] in the means where the authors demonstrate that the Tukwila architecture extends previous innovations in adaptive execution (such as mid-execution re-optimization, and rescheduling).

6. Accuracy of cost estimates

Clearly, query optimization quality depends strongly on the accuracy and the efficiency of the cost estimates [12]. There are many reasons for imprecise cost estimates 'calculated' at compile time. The statistics stored in the system's catalogue on base data may be obsolete or erroneous. Based on such estimates, predicate selectivity (join and selection) as well intermediate results size can only be more imprecise. [35] has in fact demonstrated that errors increase exponentially for complex queries. Furthermore, static optimization lacks information on both system (i.e. current state of the system) and application information which can only be obtained at run-time (e.g. available memory size, system load, host variable values).

Generally, for each database relation [12], a few basic statistics are hold: number of tuples, attribute statistics, number of physical pages, number of distinct values, and construction of histograms [1, 37, 38, 54] indicating the distribution of attribute values. In order to more accurately and more efficiently estimate these parameters, data sampling techniques [11] are intensively used. The real challenge consists in improving the estimates made with these statistics [12]. Indeed, a restricted data sample can work out an accurate histogram for only one given query and not for a large class of queries as well be desired [53]. Furthermore, they are two causes for potential imprecision in the constructed histogram: (i) the uniformity hypothesis for the distribution of values inside each histogram bucket, and (ii) the impossibility to capture the correlation between attributes.

In dynamic optimization, for a given query, even if the generation of estimates deduced from basic statistics is improved, together with information derived from base data related static context, it is still necessary to readjust the remainder of the query execution plan (i.e. the remaining fragments of the query not yet executed). This is done with real system informations obtained during execution and not priorly available at compile time. For this, Statistics Collector Operators (sampling) and Construction of System and Application informations SCO&CSA have been developed at compile time [4, 34, 39]. These operators, which need to be inserted in fragments at precise points, use recent results [38, 54] dealing with quasi-optimum data distribution and relational operator intermediate results cardinalities using specific histogram constructions. In the work of [39] spatial points to insert SCO&CSA operators are determined at compile time, as a function for join and selection predicates.

The construction cost for a histogram is far from negligible. It would be best to add these operators only where needed to improve the execution of the rest of fragments. Insertion of SCO&CSA in a fragment would not be allowed if the information produced does not concern at least one operator still to be executed. Thus, it will be interesting to evaluate whether or not improvement obtained by selective insertion is better then systematic SCO&CSA insertion. This last approach certainly generates a very high cost but enables detection and processing of all possible execution skews to appear.

7. Conclusion and open issues

Optimization of relational queries in conventional (i.e. uniprocessor) environment is recorgnized as an important, even crucial topic since the 1970's [59, 63]. Relational query optimization in a parallel environment is even more problematic due to difficulties in the development of new optimization methods and techniques taking into account the parallelism dimension, particularly in a dynamic context. In this overview, we have only considered part of the optimization problems for parallel queries in relational systems. We have not covered, for example, parallel join and short, minimizing communication costs, parallel deductive database (e.e. parallel transitive closure), and the impact of parallel architectures.

According to the state of the art on parallel query optimization, we feel several sub-themes should be more investigated: (i) *dynamic memory allocation* for shared-nothing systems [46, 52], (ii) design of new methods and efficient techniques for the *construction of quasi-optimum estimates* [1, 38], the optimization quality depends largely on the accuracy of the cost estimates and the efficiency of techniques used to calculate them, (iii) *mobile code technology* integration in the dynamic optimization of parallel and distributed queries, and (iv) *parallelization methods* for parallelizer-resource allocator due to server bottleneck.

8. References

[1] A. Aboulnaga and S. Chaudhuri, "Self-tuning Histograms: Building Histograms Without Looking of Data", *Proc. of ACM SIGMOD*, Vol. 28, ACM Press, NY, 1999, pp. 181-192.

[2] S. Bonneau and A. Hameurlain, "A Greedy Modifiable Mapping Heuristic of a SQL Query onto a Shared-Nothing Parallel Architecture" (in French), *Revue Calculateurs Parallèles,* Vol. 9, Hermès, Paris, 1997, pp. 285-304.

[3] S. Bonneau and A. Hameurlain, "Query Mapping onto a Shared-Nothing Multiprocessor Architecture: from Static to Dynamic", *Proc. of the 2 rd National Conf. on Dynamic Mapping and Load Partitioning*, Lille, France, 1998, pp. 87-91, and Technical Report (in French), No. IRIT/98-17-R, IRIT, Toulouse, 1998, 19 pages.

[4] S. Bonneau, "SQL Query Mapping onto a Shared-Nothing Multiprocessor Architecture: from Static to Dynamic", *Ph.D. Thesis* (in French), Univ. P. Sabatier/IRIT, Toulouse, 1999, 220 pages.

[5] S. Bonneau and A. Hameurlain, "Hybrid Simultaneous Scheduling and Mapping in SQL Multi-Query Parallelization", *Proc. of the 10th int'l. Conf. on Database and Expert Systems Applications DEXA*, LNCS, Vol.1677, Florence, 1999, pp. 88-99.

[6] L. Bouganim and P. Valduiriez, "Dynamic Load Balancing Hierarchical Parallel Database Systems", *Proc. of the 23rd VLDB Conf.* , Morgan Kaufmann, CA, 1996, pp. 436 - 447.

[7] L. Bouganim, O. Kapitskaia, and P. Valduriez, "Memory-Adative Scheduling for large Query", *Proc. of the 7th int'l. Conf. on Information and Knowledge Management CIKM'98*, ACM Press, NY, 1998, pp. 105-115.

[8] L. Bouganim, D. Florescu, and P. Valduiriez, "Load Balancing for Prallel Query Execution on NUMA Multiprocesors", *Distributed and Parallel Databases*, Vol. 7, Kluwer Academic, Amsterdam, 1999, pp. 99 - 121.

[9] L. Brunie and H. Kosch, "Integration of Scheduling Heuristics into Parallel Relational Query Optimization" (in French), *Revue Calculateurs Parallèles,* Vol. 9, Hermès, Paris, 1997, pp. 327-346.

[10] L. Brunie and H. Kosch, and W. Wohner, "From the modeling of parallel relational query processing to query optimization and simulation", *Parallel Processing Letters*, Vol. 8, World Scientific, Singapore, 1998, p. 2-24.

[11] S. Chaudhuri, R. Motwani, and V. Narasayya, "Random Sampling for Histgram Construction: How much is enough?", *Proc. of ACM SIGMOD*, Vol. 27, ACM Press, NY, *1998*, pp. 436-447.

[12] S. Chaudhuri, "An Overview of Query Optimization in Relational Systems", *Symposium in Principles of Database Systems PODS'98,* ACM Press, Seattle, 1998, pp. 34-43.

[13] C. Chekuri and W. Hasan, "Scheduling Problem in Parallel Query Optimization", *Symposium in Principles of Database Systems PODS* , ACM Press, NY, 1995, pp. 255-265.

[14] M.S. Chen et al., "Using Segmented Right-Deep Trees for the Execution of Pipelined Hash Joins", *Proc. of the 18th VLDB Conf.,* Morgan Kaufmann, CA, 1992, pp. 15-26.

[15] D.J. DeWitt and J. Gray, "Parallel Database Systems: The Future of High Performance Database Systems", *Communication of the ACM,* Vol. 35, ACM Press, NY, 1992, pp. 85-98.

[16] D.J. DeWitt, N. Kabra, J. Luo, J.M. Patel, and J.-B. Yu, "Client-Server Paradise", *Proc. of the 20th VLDB Conf.*, Morgan Kaufmann, CA, 1994, 558-569.

[17] S. Ganguly, W. Hasan, and R. Krishnamurthy, "Query Optimization for Parallel Execution", *Proc. of ACM SIGMOD,* Vol. 21, ACM Press, NY, 1992, pp. 9-18.

[18] S. Ganguly, A. Goel, and A. Silberschatz, "Efficient and Accurate Cost Models for Parallel Query Optimization", Proc. of PODS, ACM Press, NY, 1996, pp. 172-182.

[19] M. N. Garofalakis and Y. E. Ioannidis, "Multi-dimensional Resource Scheduling for Parallel Queries", *Proc. of ACM SIGMOD,* Vol. 25 , ACM Press, NY, 1996, pp. 365 - 376.

[20] M.N. Garofalakis and Y. E. Ioannidis, "Parallel Query Scheduling and Optimization with Time- and Space- Shared Resources", *Proc. of the 23rd VLDB Conf.,* Morgan Kaufmann, CA, 1997, pp. 296-305.

[21] G. Graefe and H. ward, "Dynamic Query Evaluation Plans", *Proc. of ACM SIGMOD,* Vol. 18, ACM Press, NY, 1989, pp. 337-388.

[22] G. Graefe and R. Cole, "Optimization of Dynamic Query Evakuation Plans", *Proc. of ACM SIGMOD*, Vol. 23, ACM Press, NY, 1994, pp. 150-160.

[23] A. Hameurlain and F. Morvan, "An Analytical Method to Allocate Processors in High Performance Parallel Execution of Recursive Queries", *Proc of the 3rd int'l. Conf., DEXA*, Springer, Berlin, 1992, pp. 44-47.

[24] A. Hameurlain and F. Morvan, "A Cost Evaluator for Parallel Database Systems", *Proc of the 6th int'l. Conf., DEXA'95*, LNCS, Vol. 978, Springer, Berlin, 1995, pp. 146-156.

[25] A. Hameurlain, F. Morvan, "Scheduling and Mapping for Parallel Execution of Extended SQL Queries", *Proc. of the 4th int'l. Conf. , CIKM'95*, ACM Press, NY, 1995, pp. 197-204.

[26] W. Hasan and R. Motwani, "Optimization Algorithms for Exploiting the Parallelism-Communication Tradeoff in Pipelined Parallelism", *Proc. of the 20th VLDB Conf.,* Morgan Kaufmann, CA, 1994, pp. 36-47.

[27] W. Hasan, "Optimization of SQL Queries for Parallel Machines", *Ph.D. Thesis*, Stanford University, 1995, 239 pages.

[28] W. Hasan, D. Floresco and P. Valduriez, "Open Issues in Parallel Query Optimization", Proc. of ACM SIGMOD, Vol. 25, ACM Press, NY, 1996, pp. 28-33.

[29] A. Helal, D. Yuan, and H El-Rewini, "Dynamic Data Reallocation for Skew Management in Shared-Nothing Parallel Databases", Distributed and Parallel Databases, Vol. 5, Kluwer Academic, Amsterdam, 1997, pp. 271-288.

[30] W. Hong and M. Stonebracker, "Optimization of Parallel Query Execution Plans in XPRS", *Proc. of the first int'l. Conf. on Parallel and Distribued Information Systems PDIS,* IEEE CS Press, CA, 1991, pp. 218-225.

[31] W. Hong, "Exploiting Inter-Operation Parallelism in XPRS", *Proc. of ACM SIGMOD,* ACM Press, NY, 1992, pp. 19-28.

[32] H.-L. Hsiao, M. S. Chen, and P. S. Yu, "On Parallel Execution of Multiple Pipelined Hash Joins", *Proc. of the 1994 ACM SIGMOD* , Vol. 23 , ACM Press, NY, pp. 185-196.

[33] K. A. Hua, C. Lee, and J.-K. Peir, "Interconnecting Shared-Everything Systems for Efficient Parallel Query Processing", Proc. of the first int'l. Conf. on Parallel Distributed Information Systems, IEEE CS. Press, CA, 1991, pp. 262-270.

[34] Z.G. Ives, D. Florescu, M. Friedman, A. Levy, and D. S. Weld, "An Adaptive Query Execution System for Data Integration", *Proc. of the 1999 ACM SIGMOD,* Vol. 28, ACM Press, NY, pp. 299-310.

[35] Y.E. Ioannidis and S. Christodoulakis, "On the Propagation of Errors in the Size of Join Results", *Proc. of the 1991 ACM SIGMOD,* Vol. 20 , ACM Pres, NY, pp. 268-277.

[36] Y. Ionnadis, R.T. Ng, K. Shim, and T. Sellis, "Parametric Query Optimization", Proc. of the 18th VLDB Conf., Morgan Kaufmann, CA, 1992, pp. 103-114.

[37] Ioannidis, Y. E., Poosala, V., "Balancing Histogram Optimality and Practicality for Query Result Size Estimation", Proc. of the 1995 ACM SIGMOD Conf., CA, pp. 233-244.

[38] H. V. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. C. Sevcik, and T. Suel, "Optimal Histograms with Quality Guarantees", *Proc. of the 24th VLDB Conf.,* Morgan Kaufmann, CA, 1998, pp. 275-286.

[39] N. Kabra and D. - J. DeWitt, "Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans", *Proc. of the 1998 ACM SIGMOD,* Vol. 27, ACM Press, NY, pp. 106-117.

[40] N. Kabra and D. - J. DeWitt, "OPT++: An Object-Oriented Implementation for Extensible Database Query Optimization", *VLDB Journal,* Vol. 8, Springer, Berlin, 1999, pp. 55-78.

[41] M.F. Khan, R. Paul, I. Ahmed, and A. Ghafoor, "Intensive Data Management in Parallel Systems: A Survey", *Distributed and Parallel Databases,* Vol. 7, Kluwer Academic, Boston, 1999, pp. 383-414.

[42] M. Kitsuregawa and Y. Ogawa, "Bucket Spreading Parallel Hash: A New, Robust, Parallel Hash Join Method for Data Skew in the Super Database Computer", *Proc. of the 16th VLDB,* Morgan Kaufmann, CA, 1990, pp. 210-221.

[43] H. Kosch, "Exploiting Serialized Bushy Trees for Parallel Relational Query Optimization", *Ph.D. Thesis,* Ecole Normale Supérieure de Lyon LIP , Lyon, 1997, 189 pages.

[44] H. Kosch, "Managing the operator ordering problem in parallel databases", *Future Generation Computer Systems,* Vol.14, Elsevier, Amsterdam, 1999.

[45] J.-H. Lee, D.-H. Kim and C.-W. Chung , "Multi-dimensional Selectivity Estimation Using Comrpressed Histogram Information", *Proc. of the 1999 ACM SIGMOD,* Vol. 28, ACM Press, NY, pp. 205-214.

[46] T. Leighton and E.J. Schwabe, "Efficient Algorithms for Dynamic Allocation of Distributed Memory", Algoritmica, Vol. 24, Springer, NY, 1999, pp. 139-171.

[47] Y. Lin, W. Sun, N.D. Rishe, and X. Xiang, "A Hybrid Estimator for Selectivity Estimation", *IEEE TKDE,* Vol. 11, IEEE CS Press, 1999, pp. 338-354.

[48] H. Lu and K.L. Tan, "Dynamic and Load-Balanced Task-Oriented Database Query Processing in Parallel Systems", *Proc. 3rd Int'l Conf. Extending Data Base Technology* , 1992, pp. 357-372.

[49] H. Lu, B.C. Ooi, and K.L. Tan, *Query Processing in Parallel Relational Database Systems,* IEEE CS Press, CA, 1994.

[50] M. Mehta and D.-J. DeWitt, "Managing Intra-Operator Parallelism in Parallel Database Systems", *Proc. of the 21th Int'l. Conf. on VLDB,* Morgan Kaufmann, CA, 1995, pp. 382-394.

[51] M. Mehta and D.-J. DeWitt, "Data Placement in Shared-Nothing Parallel Database Systems", *The VLDB Journal,* Vol.6, Springer, Berlin, 1997, pp. 53-72.

[52] B. Nag and D.-J. DeWitt., "Memory Allocation Strategies for Complex Decision Support Queries", *Proc. of the 7th Int'l. Conf. on Information and Knowledge Management CIKM'98,* ACM Press, NY, 1998, p. 116-123.

[53] Piatetsky-Shapiro, G., Connel, C., "Accurate Estimation of the Number of Tuples Satisfying a Condition", Proc. of ACM SIGMOD, ACM Press, NY, 1984, pp. 256-276.

[54] V. Poosala and Y.E. Ioannidis, "Estimation of Query-Result Distribution and its Application in Parallel-Join Load Balancing", *Proc. of the 23rd VLDB Conf.,* Morgan Kaufmann, CA, 1996, pp. 448-459.

[55] E. Rahm and R. Marek, "Analysis of Dynamic Load Balncing Strategies for Parallel Shared Nothing Database Systems", *Proc. of the 19 th VLDB Conf.,* Morgan Kaufmann, CA, 1993, pp.182-193.

[56] E. Rahm and R. Marek, "Dynamic Multi-Resource Load Balancing in Parallel Database Systems", *Proc. of the 21st VLDB Conf.,* Morgan Kaufmannn, CA, 1995, pp. 395-406.

[57] D. Schneider and D.-J. DeWitt, "A Performance Evaluation of four Parallel Join Algorithms in a Shared-Nothing Multiprocessor Environment", *Proc. of the 1989 ACM SIGMOD,* ACM Press, NY, pp. 110-121.

[58] D. Schneider and D.J. DeWitt, "Tradeoffs in Processing Complex Join Queries via Hashing in Multiprocessor Database Machines", *Proc. of the 16th VLDB Conf.,* Morgan Kaufmann, CA, 1990, pp. 469-480.

[59] P. G. Selinger, M. Astrshan, D. Chamberlin, R. Lorie, and T. Price, "Access Path Selection in a Relational Database Management System", *Proc. of the 1979 ACM SIGMOD,* ACM Press, NY, pp. 23-34.

[60] K.-L. Tan and H. Lu, "On Resource Scheduling of Multi-Join Queries in Parallel Database Systems", *Information Processing Letters,* Vol. 48, Elsevier, Amsterdam, 1993, pp. 189-195.

[61] P. Valduriez, "Parallel Database Systems: Open Problems and News Issues", *Distributed and Parallel Databases,* Vol. 1, Kluwer Academic, Boston, 1993, pp. 137-165.

[62] C.B. Walton and A.G. Dale, and R.M. Jenevin, "A Taxonomy and Performance Model of Data Skew Effects in Parallel Join", *Proc. of the 17th int'l. Conf. on VLDB,* Morgan Kaufmann, CA, 1991, pp. 537-548.

[63] E. Wong, and, K. Youssefi, "Decomposition: A Strategy for Query Processing", *ACM Transaction on Database Systems,* Vol. 1, ACM Press, CA, 1976, pp. 223-241.

[64] M. Ziane, M. Zait, and P. Borlat-Salamet, "Parallel Query Processing in DBS3", 2nd Intl. Conf. on Parallel and Distribued Information Systems, CA, IEEE CS Press, 1993, pp. 93-102