Proceedings of the Federated Conference on Computer Science and Information Systems pp. 1383–1387
DOI: 10.15439/2016F123
ACSIS, Vol. 8. ISSN 2300-5963

Speculative Query Execution in Relational Databases with Graph Modelling.

Anna Sasak-Okoń
University of Maria Curie-Skłodowska
in Lublin
Pl. Marii Curie-Skłodowskiej 5, 20-031 Lublin, Poland
Email: anna.sasak@umcs.pl

***Abstract*—In computer architecture, speculative execution is the process of executing instructions ahead of their normal schedule[1]. Grama et al.[2] introduce the concept of speculative decomposition as a possibility to execute one or more of possible branches in parallel with computation which are expected to determine the branch choice. The following paper introduces the method of speculative query execution in relational databases. Query queue can be seen as a line of sequential instructions and thus changing their order can result in some errors. Author introduce a middleware called the Speculative Layer which, based on a specific graph representation, executes some additional Speculative Queries. Results of those Speculative Queries can be used while executing queries from the queue providing a befit which is a shorter response time. The paper describes the process of graph modelling for groups of queries in order to initiate speculative computations, metrics used to evaluate Speculative Queries and experimental results for a test database and a group of input queries.**

I. Introduction

ORIGINS of the speculative execution are the early works of branch prediction[3][4]. The sequential semantics imposes a certain order in which instructions should be loaded, decoded, executed and ended[5]. Code branches, usually dependent on some logical conditions, disturb the fluency of loading and executing, causing delays. As an attempt to prevent delays there were experiments to predict a branch direction and to execute an instruction or a group of instructions in advance.

In general, there are three types of Thread Level Speculations (TLS)[6]:

- Control Speculation – origins from branch prediction strategy. The assumptions could be made based on some static (e.g. op codes) or dynamic values [7][8].
- Data Dependence Speculation – If two instructions are fully independent, only then the parallel execution is possible. Before memory access instructions are executed, the addresses they refer to are often undetermined. To prevent data load from an address where store should be executed earlier, a certain secure mechanism should be introduced[8].
- Data Value Speculationis – is expected to prevent data dependency with the value prediction mechanisms which allow to propagate data values to succeeding instructions in advance.

II. Related Work

There is already much research done around the world in adopting speculative execution in database computations.

Polyzotis N. and Ioannidis Y[10]. introduce speculation as a parallel, intelligent technique of query processing assistance. Exploiting idle time[11] of the system the application processes some asynchronous database manipulations which in case of success would be beneficial for the final query.

Barish G. and Knoblock C.A.[12][13] in order to overcome the limits imposed by binding patterns between data sources propose mechanisms of applying speculative execution for Information Gathering Plans. The general process of speculative execution involves issuing operations ahead of their normal schedule based on data (hints) received earlier in the plan.

Hristidis V. and Papakonstantinou Y[14] analyse speculative computations for ranked queries. Authors create a speculative version of a ranking algorithm which in case of a slower data source assumes speculatively that there are no tuples satisfying the preference function and thus can return top-N results faster but with some inaccuracies.

Reddy P., Kitsuregawa M.[15], Ragunathan T., Krishna R.P.[16][17] deal with speculative execution for transaction protocols in database systems. They introduce the speculative protocol (SL). With SL the waiting transaction is able to access locked data as soon as blocking transaction produces its images.

III. The Speculative Layer

An inspiration for this experiment is an idea of speculative execution briefly described in the previous section. Authors proposes a speculative execution mechanism for relational databases executing SQL queries of accepted structure which are CQAC queries with additional IN and LIKE operators. What is important, an analysed database must show a specific use template. Databases which suit our interests usually have to execute similar queries from different users. What's more, data modifications are rare and usually concentrate around some fixed points.

A queue of queries awaiting for execution, called the input queries, presents an interesting analogy to the sequential order of instructions. The consecutive queries can, like sequential instructions, show some dependencies. On the other hand, carefully identified similarities allow to use some of the results many times[18][19][20].

A model described above is implemented as an additional middleware between users and DBMS called the Speculative Layer, which dynamically supports execution of input queries. The Speculative Layer, based on precise Speculative Analysis, creates a subset of data in RAM called further a Speculative DB. The data from the Speculative DB used while executing an input query improves system throughput and shortens users waiting time.

All actions of the Speculative Layer are controlled by the main worker thread called the Manager. In each step N input queries are analysed. This group of input queries is called the Window of Speculations. Based on those analysis, supported by a specific graph representation described in Sections IV and V, Manager assigns tasks to K Worker Threads.

In particular Manager implements the following functions of the Speculative Layer:

1) System Start.
   All initial actions required for the first run of the Speculative Layer. In particular graph representations are created for N input queries from the Window of Speculation. Next, those representations are combined to create the Queries Multigraph ready for the Speculative Analysis.
2) Nonspeculative Query Execution.
   Process of executing the first input query from the Window of Speculation, called the Nonspeculative Query. If there are Executed Speculative Queries assigned to the Nonspeculative Query, then it must by modified so it would use those results. If there are more than one Executed Speculative Query assigned, then the choice which to use is based on the values of the defined metrics - Horizontal an Vertical Selectivity. Vertical Selectivity models the reduction of the number of columns while the Horizontal Selectivity approximates of number of records returned by the Speculative Query.
3) Speculative Analysis.
   Process of the Queries Multigraph analysis which identifies speculation points and generates Awaiting Speculative Queries. The other result of the Speculative Analyses is a Speculative Queries Multigraph i.e. Queries Multigraph with additional speculative edges representing points and types of speculations.
4) Window of Speculation Move.
   After the Nonspeculative Query is executed and its results are returned to the user the Window of Speculations moves. It means that the representation of executed Nonspeculative Query in the Queries Multigraph is replaced by the representation of the next input query from the queue.
5) Speculative Query Execution.
   If there are idle Worker Threads then, if it is possible, they should be assigned available Speculative Queries from the Awaiting Speculative Queries List, according to the values of aforementioned metrics. The highest execution priority should have those queries which provide the highest potential reduction of records or/and can be used by the most of Input Queries.
6) Executed Speculative Query Assignment.
   After an Awaiting Speculative Query is executed and becomes an Executed Speculative Query, it has to be assigned to the specific Input Query/Queries from the Window of Speculations, which marks the possibility to use its results.
7) Speculative DB Refreshment.
   When the Speculative DB reaches its maximum size, it has to be reduced. The reduction process consists in removing the results of chosen Executed Speculative Queries based on its characteristics. First to remove are always those queries with the highest Vertical and Horizontal Selectivity and those which are used the least often.

IV. Query Graph

A. CQAC queries

Each accepted CQAC query is represented by its Query Graph $G_Q(V_Q, E_Q)$. Graph creation rules follow the example of [22][23] works, and are as follows. Each Query Graph Vertex is one of three types:

- Relation Vertex (R_i) – one for each relation,
- Attribute Vertex (A^i_j) – one for each attribute,
- Value Vertex $(\Omega) = \{Val^i_j | A^i_j\}$ – one for each value or set of values.

Each Query Graph Edge is one of the following types:

- Membership Edge – $e_\mu : R_i \overset{\mu}{-} A^i_j$ – one between relation R_i and each of its attributes A^i_j from SELECT clause,
- Predicate Edge – $e_\theta : A^i_j \overset{\theta}{-} \{Val^i_j | A^m_k\}$ – one for each predicate of WHERE clause $A^i_j \theta \Omega$, where θ is one of accepted operators. Ω is a single value or a set of values (Val^i_j) or an attribute (A^m_k) for JOIN condition.
- Selection Edge – $e_\sigma : R_i \overset{\sigma}{-} A^i_j$ – one for each predicate of WHERE clause $A^i_j \theta \Omega$, where θ is one of accepted operators. Ω is a set of values (Val^i_j) or an attribute (A^m_k) for JOIN condition.

B. Embedded queries

Each embedded query q_m is represented by its own query graph joined with its parent query Q graph in the following way – for each predicate $A^i_j \theta A^m_k$ where $A^i_j \in$ (Q WHERE clause) and $A^m_k \in$ (q_m SELECT clause), there is a predicate edge between A^i_j and A^m_k.

C. Modifying queries

Next to SELECT queries there are also modifying queries which are accepted by the Speculative Layer and thus need to have a proper graph representation. For each type of modifying query there is another edge type representing a possible change in the database state:

- DELETE: $e_\delta : R_i \overset{\delta}{-} A^i_j$ and $e_\delta : A^i_j \overset{\upsilon\theta}{-} \Omega$ where θ is one of accepted operators and Ω is a set of values,

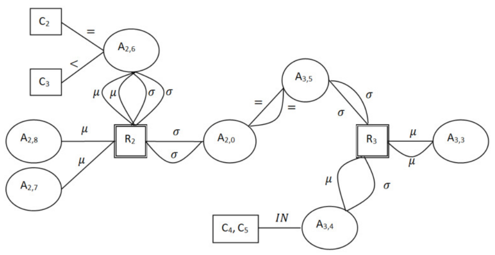

Fig. 1. Queries Multigraph

- INSERT: $e_\eta : R_i \overset{\eta}{-} R_i$
- UPDATE: $e_\upsilon : R_i \overset{\delta}{-} A^i_j$ and $e_\upsilon : A^i_j \overset{\upsilon\theta}{-} \Omega$ where θ is one of accepted operators and Ω is a set of values.

V. Queries Multigraph

To represent a group of queries with one graph some additional rules have to be defined. Such graph $G_s(V_s, E_s)$ will be called the Queries Multigraph or QM. QM Vertices set is an union of Vertices of all component query graphs: $V_s = V_{q1} \cup V_{q2} \cup \ldots \cup V_{qn}$. QM Edges set is a multiset of all component query graphs edges: $E_s = E_{q1} + E_{q2} + \ldots + E_{qn}$. This way multiple edges of the same type are allowed. It is important for the Speculative Analysis process raising the importance of some edge connections. Fig.1 presents the Queries Multigraph representing three following component queries:

- SELECT $A_{3,3}, A_{2,6}, A_{2,7}$ FROM R_2, R_3 WHERE $A_{2,0} = A_{3,5}$ AND $A_{2,6} = C_2$
- SELECT $A_{2,6}, A_{2,8}$ FROM R_2, R_3 WHERE $A_{2,0} = A_{3,5}$ AND $R_2.A_{2,6} < C_3$
- SELECT $A_{3,3}$ FROM R_3 WHERE $R_3.A_3, 4$ IN (C_4, C_5)

VI. Types of Speculative Queries

The process of Speculative Analysis is expected to determine a set of Speculative Edges. Those edges represent different strategies of creating Speculative Queries. Based on the results usage, there are three types of Speculative Edges which represent three types of Speculative Queries:

- Speculative Parameter – those edges/queries relate to the presence of embedded queries. Due to separating an embedded query as a Speculative Query, it is possible to use its results as a parameter in a parent query. The Speculative Parameter Speculation is identified the moment the query with an embedded select enters the Window of Speculation. An embedded query as a whole is added to the head of the Awaiting Speculative Queries List (Q^i_{PS}). As a consequence those queries are always first to be executed by the Worker Thread.
- Speculative Data – the aim of those speculative queries is to obtain and save in the Speculative DB a specific subset of records or/and attributes of a relation. The main goal is to create this subset so as it could be used while executing as many input queries as possible. Speculative Data queries are the most frequent in the group of identified speculations.
- Speculative State – those edges/queries relate to the presence of modifying queries. If there is a modifying query in the Window of Speculation then both Executed and Awaiting Speculative Queries are in danger of processing invalid data. Speculative State edges are referring to the modifying queries represented by $e_\delta, e_\eta, e_\upsilon$ edges. If a modifying query has a number K in an input queries queue, then all succeeding queries (K+) are in danger of processing invalid data.

The Fig. 2 presents an example of Queries Multigraph with Speculative Data and Speculative State edges.

VII. Experimental Results

The Speculative Layer was implemented with C++ and Visual Studio 2013 with Pthread library and SQLite 3.8.11.1. Experimental results were obtained in Windows 8.1 64b with Intel Core i7-3930K and 8GB RAM.

A. Test Database and Test Input Queries

The Database used for experiments was generated with the TPC[21] data and structure generator. It consists of 8 relations storing 1GB data. It represents 150 000 orders from 150 000 customers which include chosen from 200 000 products delivered by 10 000 suppliers. Such database is a fine example of a medium sized Internet store. Eight SQL Query Templates were prepared and used to generate a set of Input Queries executed with the Speculative Layer.

B. Window of Speculation Size and the Number of Speculative Threads

At the beginning the series of experiments were conducted to determine the size of Window of Speculation and the number of Speculative Threads for which the experiments would continue. The size of the Window of Speculation stands for the number of input queries represented by the Query Multigraph and thus it determines the number of generated Awaiting Speculative Queries. Fig.3 presents how the Size of Window of Speculation affects the number of generated Awaiting Speculative Queries.

The number of executed Speculative Queries depends on the number of active Speculative Threads and not on the size of the Window of Speculation itself. In the Fig.4, two series of data are presented. First one, marked with black squares, presents the number of Executed Speculative Queries for the number of active Speculative Threads. The second one, marked with

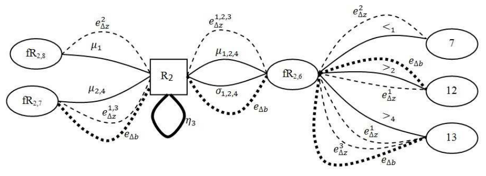

Fig. 2. Multigraph with Speculative Data and Speculative State edges

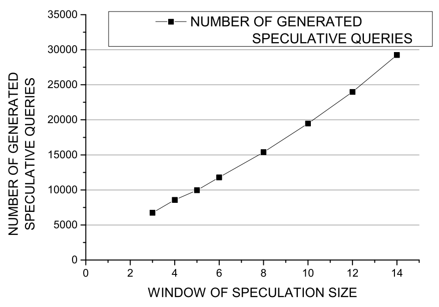

Fig. 3. The number of generated Awaiting Speculative Queries.

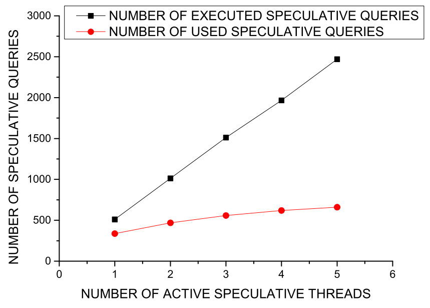

Fig. 4. The number of executed and used Speculative Queries.

red dots, presents the number of Used Speculative Queries for the number of active Speculative Thread. The experiment was carried for the Window of Speculation Size = 5.

Fig. 3 shows almost linear dependency between the Size of the Window of Speculation and the number of generated Awaiting Speculative Queries. Fig. 4 also presents almost linear dependency between the number of executed Speculative Queries and the number of active Speculative Threads. On the other hand the number of Used Speculative Queries hardly changes for the number of threads 3 and more and thus the percent of Used Speculative Queries is decreasing. Based on those observations it was decided that the further experiments would be carried for the Window of Speculation Size=5. The experimental results are presented for the set of 1000 input queries generated from Templates T1-T8. The size of Speculative DB is 700MB RAM.

C. Query Execution Times

Fig. 5 presents the reduction of average execution time for input queries of each Template. First column represents the sequential execution time when there were no active Speculative Threads. The following columns represent execution times obtained for each query Template for 1 to 5 active Speculative Threads, which execute Speculative Queries. It appears that the highest execution time reduction was obtained by initiating the first Speculative Thread (up to 55% execution time reduction for Template 1). Further improvement, up to 20% brings the second active Speculative Thread. Activating

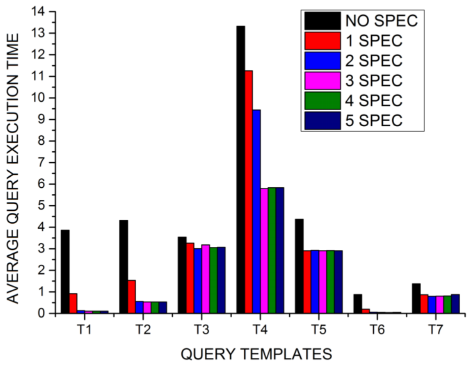

Fig. 5. The average Execution Time for Each Template for 0-5 Active Speculative Threads.

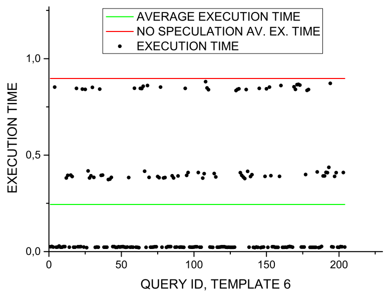

Fig. 6. Query Execution Times for Template 6.

more than two Speculative Threads doesn't affect the Average Execution Time.

Fig. 6 presents execution times of input queries for Template 6, with the Window of Speculation Size=5 and 2 active Speculative Worker threads. In the picture there are two additional lines presented showing the average execution time with (green line) and without (red line) speculative execution. Template 6 was chosen for presentation as its queries show an interesting behaviour. There is a clear division for three groups of input queries. First group are the input queries executed without the opportunity to use results of Executed Speculative Query. They have the longest execution times and thus are located close to the red line. The remaining two groups are input queries which were able to use the results of Executed Speculative Queries, however, the obtained execution times vary significantly. It turned out the group with the lowest execution times had an opportunity to use results of the Speculative Query with the Horizontal Selectivity equal to 0,01. The rest of them had to use the results of Speculative Queries with the Horizontal Selectivity equal to 0,9 which are almost full copy of the original ORDERS relation.

Fig. 7 presents how many Input Queries of each Template were executed using results of the Executed Speculative Query. It is expressed as a proportional dependency where each column stands for 100% of Input Queries of each Template. As you can see Templates T3 and T4 are the least responsive

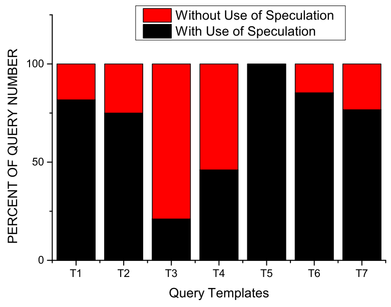

Fig. 7. The number of Input Queries which used the Executed Speculative Query Results as a proportional relation to the total number of Input Queries of each Template.

to Speculation and thus the average execution time reduction of those templates (Fig.5) was also the lowest. The reason is that those queries has low occurrence density (10%) in the Input Query Set. As a consequence its multigraph vertices has low Selection Degrees and thus they are rarely chosen to be executed by Speculative Threads. What's more, in both those Templates, T3 and T4, IN and LIKE operators are dominating. Those operators refer to the narrow subgroups of attribute values, making it especially difficult to generate Speculative Queries useful for more than one query from the Window of Speculation.

VIII. Conclusion

The following paper presents a model of Speculative Execution to support SQL query execution in relative databases. The Speculative Layer executing Speculative Queries is carefully described. In particular the details of an adopted query graph representation are presented. Experimental results obtained for the test database and a group of 1000 input queries are very promising. In case of Template 5, 100% of input queries were execute using the results of executed Speculative Query. Templates: 1, 2, 6 and 7 show around 75% and above execution with Speculative Query. All groups of queries show its execution time reduction: from 10%(Template 3) to 70%(Template 6). Further work should concentrate on the improvement of the number of input queries executed with the Speculative Query results (especially for Template 3 and 4). Worth consideration is also allowing more flexible structure of accepted queries and intensifying the number of modifying queries which would require more sophisticated speculation validation methods.

References

[1] D. Kaeli, P. Yew, "Speculative Execution in High Performance Computer Architectures," Chapman Hall/CRC, 2005, ISBN:978-1-584-88447-7.

[2] A. Grama, A. Gupta, G. Karypis, V. Kumar, "Introduction to Parallel Computing (Second Edition)," Addison Wesley, 2003, ISBN: 978-0-201-64865-2.

[3] E. A. Jr. Liles, B. Wilner, "Branch prediction mechanism.," IBM Technical Disclosure Bulletin, 1979, Vol.22(7), p. 3013-3016.

[4] J. E. Smith, "A study of branch prediction strategies.," ISCA 98 Conference Proceedings, 1998, New York, p.202-215, http://dx.doi.org/10.1145/285930.285980.

[5] D. Padua, "Encyclopedia of Parallel Computing A-D.," Springer, 2011, ISBN: 978-0-387-09765-7.

[6] A. Kejariwal, X. Tian, W. Li, M. Girkar, S. Kozhukhov, H. Saito, U. Banerjee, A. Nicolau, A.V. Veidenbaum, C.D. Polychronopoulos, "On the performance potential of different types of speculative thread-level parallelism.," International Conference on Supercomputing Proceedings., 2006, Cairns, p.24, http://dx.doi.org/10.1145/1183401.1183407.

[7] J. Šilc, T. Ungerer,B. Robič, "Dynamic branch prediction and control speculation.," Int. Journal of High Performance Systems Architecture, 2007, Vol.1(1), p.2-13, http://dx.doi.org/10.1504/IJHPSA.2007.013287.

[8] D. Kaeli, P. Yew, "Speculative Execution in High Performance Computer Architectures.," Chapman & Hall,CRC, 2005, ISBN:978-1-584-88447-7.

[9] S. T. Pan, K. So, J. T. Rahmeh, "Improving the accuracy of dynamic branch prediction using branch correlation.," International Conference on Architectural Support for Programming Languages and Operating Systems, 1992, Boston, p.76-84, http://dx.doi.org/10.1145/143371.143490.

[10] N. Polyzotis, Y. Ioannidis, "Speculative query processing," CIDR Conference Proceedings, Asilomar, 2003, p.1-12, http://dx.doi.org/10.1.1.11.8541.

[11] R. M. Karp, R. E. Miller, S. Winograd, "The Organization of Computations for Uniform Recurrence Equations," Journal of the ACM, 1967, Vol.14(3): p.563-590, http://dx.doi.org/10.1145/321406.321418.

[12] G. Barish, C.A. Knoblock, "Speculative Plan Execution for Information Gathering," Artificial Intelligence, 2008, Vol.172(4-5), p.413-453, http://dx.doi.org/10.1016/j.artint.2007.08.002.

[13] G. Barish, C.A. Knoblock, "Speculative Execution for Information Gathering Plans," AIPS Conference Proceedings, Toulouse, 2002, p.184-193, http://dx.doi.org/10.1.1.11.3505.

[14] V. Hristidis, Y. Papakonstantinou, "Algorithms and Applications for answering Ranked Queries using Ranked Views," The VLDB Journal, 2004, Vol.13(1), p.49-70, http://dx.doi.org/10.1007/s00778-003-0099-8.

[15] P.K. Reddy, M. Kitsuregawa, "Speculative locking Protocols to Improve Performance for Distributed Database Systems," IEEE Transactions on Knowledge and Data Engineering, 2004, Vol.16(2), p.154-169, http://dx.doi.org/10.1109/TKDE.2004.1269595.

[16] T. Ragunathan, R. P. Krishna, "Performance Enhancement of Read-only Transactions Using Speculative Locking Protocol," IRISS - Sixth Annual Inter Research Institute Student Seminar in Computer Science, Hyderabad, 2007.

[17] T. Ragunathan T, R. P. Krishna, "Improving the performance of Read-only Transactions through Speculation," Databases in Networked Information System, 2007, Vol.4777, p.467-474, http://dx.doi.org/10.1007/978-3-540-75512-8_15.

[18] A. Sasak-Okoń, M. Brzuszek, Speculative execution plan for multiple query execution systems, Annales UMCS Informatica, 2010, Vol 10(2), p.41-50,

[19] A. Sasak-Okoń, M.Brzuszek, The example of speculative execution for multiple query execution systems, Metody Informatyki Stosowanej, 2011, Vol.3(28), p.157-166, ISSN 1898-5297.

[20] A. Sasak-Okoń, M. Brzuszek, Graph modeling as a support technique for speculative computations in multiple query execution systems, Data Analysis Selected Problems, 2013, p.55-68, ISBN 978-83-7518-602-4.

[21] TPC benchmarks, http://www.tpc.org/tpch/default.asp, 2015.

[22] G. Koutrika, A. Simitsis, Y. Ioannidis, "Conversational Databases: Explaining Structured Queries to Users", 2009, Technical Report Stanford InfoLab.

[23] G. Koutrika, A. Simitsis, Y. Ioannidis, "Explaining Structured Queries in Natural Language.", ICDE Conference Proceedings, Long Beach, 2010, p. 333-344, http://dx.doi.org/10.1109/ICDE.2010.5447824.