IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 5, May 2016

Computing Shortest Path for Graph Using FEM Framework

Piyush Kulkarni[1], Kapil Vyas[2]

PG Student, Computer Science and Engineering, BM College of Technology, Indore (MP), India[1]

Assistant Professor, Computer Science and Engineering, BM College of Technology, Indore (MP), India[2]

Abstract: This paper focus on computing shortest path as finding the distance between two vertices in graph or tree. Graph data can use in many domains like a social network and in knowledge graph. This graph search includes sub-graph. The problem of finding the shortest path between two nodes can be solved using the salesman traveling path, minimal spanning tree, and the likewise. The problem occurred with the graph based searching when a graph is too big to fit in memory and for that, it uses the external memory. The disk-based method has some limitations when graph exceeds its size. In this paper, we are analyzing the shortest path for efficient relational approaches to graph search queries. For this, we use three relational operator based-on which we introduce the framework for bridge the gap between graph operation and relational operator. We show the new feature of SQL such as merge statement and windows function to improve the performance of FEM framework. To avoid extra indexing overhead and improve scalability and performance, we propose an edge weight aware graph partitioned schema and design bi-directional restrictive BFS (breadth-first-search). The final experimental results illustrate our relational approach with optimization strategies can achieve high performance and scalability.

Keyword: graph, shortest path, Relational database, graph search queries, graph indexing.

I. INTRODUCTION

Today's world growth of graph increases rapidly, graph search faces more challenges. Graph search is more common in graph applications. Graph search purse for a specific purpose such as shortest path between two nodes, minimum spanning tree etc. As a size of graph increases, it does not fit into main memory so existing approaches to graph search must be reexamined. On external disk memory, I/O is the key factor for graph operations. As graph size increases, existing disk-based methods provide limited support for graph based quires. Neo4J is one who can a store large graph in a database and provide operation such as shortest path discover to end users and graph traversal. The performance of graph based systems should be continuously improved, as graph database systems have to implements complex component including query evaluation, query optimization, storage etc. MapReduce framework and its open source implementation Hadoop can process large graph stored on distributed file system.

Relation database (RDB) provides support for graph search. RDB plays a key role in information systems. RDB and graph database management have many same functionalities such as storage, optimization etc. RDB also managing a complex database like XML data. RDB can support graph queries such as BFS and reachability query .The extension of RDB to graph search queries is useful when both graph and relational operation are needed. It requires a substantial effort to support graph search query in RDB context. First, queries of graph search are various forms. It is not possible to implement each query. We need to find a mechanism for evaluation of graph search in relational context. Second, there is a symbolic mismatch between relational operations and graph operation which effects on graph search. Graph operation follows node-at-time fashion whereas relational operations follow set-at-time fashion. This paper focuses on shortest path discovery for two reasons. First, search the shortest path that provides a key role in many applications like reveal relationship between two individual in a social network. Second it represents query which has similar evaluation pattern like other search queries.

II. PROPOSED SYSTEM

In a graph, various types of algorithm are present for the finding the shortest path such as best first search, breadth first search, Dijkstra's algorithm. Dijkstra's algorithm and bidirectional restrictive BFS algorithm is used to find the shortest path with the help of FEM framework using a relational database.

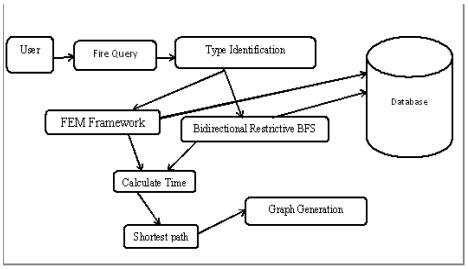


Fig -1: System Architecture

The system mainly divided into following parts

Query processing

We use two SQL feature window function and merge statement which is supported by many database systems

DOI 10.17148/IJARCCE.2016.55108

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 5, May 2016

like oracle, SQL server. Window function returns an aggregate of each tuple in a set. The merge statement adds new tuples and update existing one in a table from a source table.

FEM Framework

To find the shortest path between a source node and targeted node, we use Dijkstra's algorithm. There are three basic operations in graph search select, expand and merge respectively. In this, we first initialize a table with source node then start iteration to find the shortest path between a source node and targeted node. We locate frontier node and perform expansion and merge newly expanded node into visited nodes.

Bidirectional Restrictive BFS

Bidirectional allows forward expansion and backward expansion. In bidirectional restrictive BFS, we allow multiple a node be frontier node and search from this frontier node along their partial incident edge first. The expansion can be done in set-at-a-time fashion and visited node to be re-expanded as compared to BFS. We also use partitioning table strategy to handle large graphs.

ALGORITHMS

For finding the shortest path, we used two algorithms.

1) Dijkstra's algorithm for shortest path discovery
2) Bi-Directional Restrictive BFS on Partitioned Tables

The Dijkstra's algorithm for shortest path discovery as follow

Input: Source node s, target node t, Graph G= (V, E)

Output: Shortest path between s and t.

1) Initiaze all visited node
2) While true do
3) Locate mid value , ID for next frontier node using SQL
4) Expand path using query operator in SQL with mid value
5) When tuples is Zero then break
6) Finalize frontier node where identifier= mid value
7) If result exists then break
8) Find edges in shortest path along with preprocessor source node
9) Return the shortest path

The Bi-Directional Restrictive BFS on Partitioned Tables as follow

Input: source node s, target node t .partitioned tables

Output: shortest path between s and t

1) Initialize TA^f with node s and TA^b with node t
2) minCost assume infinity
3) ~~i=1, j =1 l_i^f =0, l_j^b=0 , n^f=1, n^b=1~~ ← ←
4) While $l_i^f + l_j^b$<= minCost do
5) ifn^f<= n^b then
6) Expand the path
7) n^f = no. of affected tuples
8) Compute l_i^f using minimum distance in forward searching
9) I i+1Else ←
10) Similar action from line 7 to 10 for backward expansion
11) Locate minimum distance
12) Make verification of minimum distance
13) Compute minimum distance.
14) Locate node in shortest path nid
15) Find sub-path sp0 from s to nid
16) Find sub-path sp1 from nid to t
17) Return sp0+sp1;

MATHEMATICAL MODEL

The complete system S can be represented in terms of input, output and functions.

S = {I, O, F}

I = Set of inputs

O = Set of output

F = Set of functions

I= {I1, I2}

I1 = Graph

I2 = User Query

O = {O1, O2, O3}

O1 = Shortest path between two node

O2 = Average time cost

F = {F1, F2, F3, F4, F5, F6, F7, F8, F9 }

F1 = Graph generation

F2 = Query Analysis

F3 = Type Identification

F4 = Query preparation

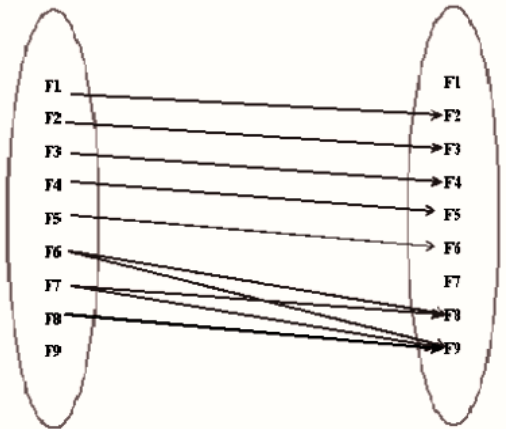
F5 = SQL in Path Finding

F6 = Query Expansion

F7 = Distance Calculation

F8 = Dijkstra's algorithm

F9 = Bi-directional Restrictive BFS



III. EXPERIMENTAL RESULT

Dijkstra's Algorithm start at source node S, span all node reachable from S, then next closest node and so on. When FEM framework is used along with it, one node will be selected as a frontier node and search in one direction. In restrictive BFS, we allow multiple nodes be frontier node and search from this frontier node. WE find a path using a window function and merge statement and traditional function such as aggregate and insert/update. The result

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 5, May 2016

shows that the system performance increases with a window function and merge statement. When we using FEM framework, time requires for finding shortest path is decreased.

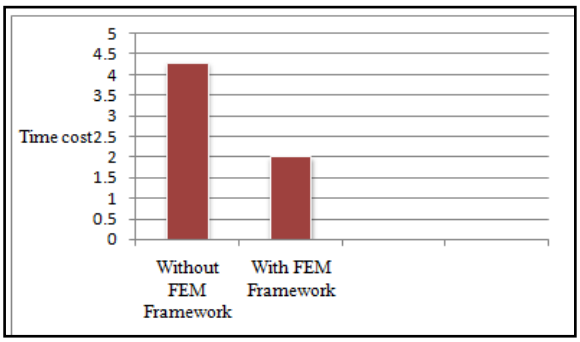


Chart 1: Time Cost analysis for with and without FEM Framework

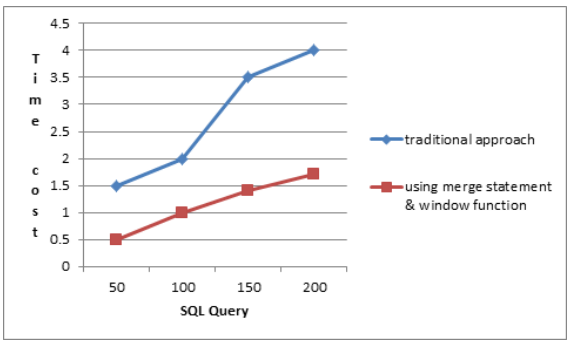


Chart 2: Time vs SQL Query

We can also see our bi-directional restrictive BFS on weight aware partition edge table consume more time than Dijkstra's algorithm.

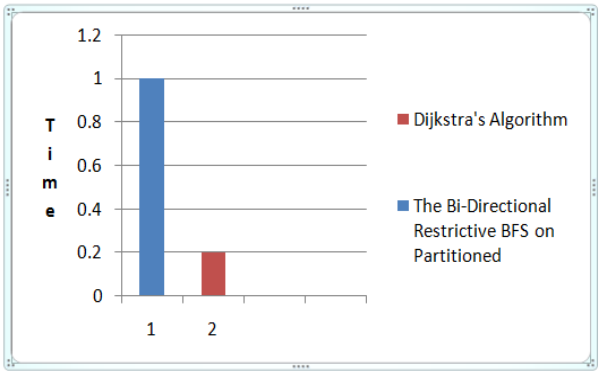


Chart 3:Time comparison between Dijkstra's and Bi-directional Restrictive BFS

IV. CONCLUSIONS

This paper proposes FEM framework to find a short path between a source node and targeted node. It also introduces new SQL feature such as merge statement and windows function to improve the performance of FEM framework. The bidirectional restrictive search and graph table is partitioned improves the scalability and performance of FEM Framework. The proposed system requires less execution time and gives a better result than bidirectional search.

REFERENCES

[1]. B. Bahmani, K. Chakrabarti, and D. Xin, "Fast Personalized Pagerank on Mapreduce," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '11), pp. 973-984, 2011.
[2]. S. Trißl and U. Leser, "Fast and Practical Indexing and Querying of Very Large Graphs," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD'07), pp. 845-856, 2007.
[3]. S. Srihari, S. Chandrashekar, and S. Parthasarathy, "A Framework for SQL-Based Mining of Large Graphs on Relational Databases," Proc. 14th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining—Volume Part II (PAKDD'10), pp. 160-167, 2010.
[4]. B. Zou, X. Ma, B. Kemme, G. Newton, and D. Precup, "Data Mining Using Relational Database Management Systems," Proc. 10th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining ('06), pp. 657-667, 2006.
[5]. M. Potamias, F. Bonchi, C. Castillo, and A. Gionis, "Fast Shortest Path Distance Estimation in Large Networks," Proc. Int'l Conf. Information and Knowledge Management (CIKM'09), pp. 453-470, 2009.
[6]. F. Tian, B. Reinwald, H. Pirahesh, T. Mayr, and J. Myllymaki, "Implementing a Scalable XML Publish/Subscribe System Using a Relational Database System," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '04), pp. 479-490, 2004.
[7]. D. Wagner and T. Willhalm, "Speed-Up Techniques for Shortest-Path Computations," Proc. 24th Ann. Conf. Theoretical Aspects of Computer Science (STACS '07), pp. 23-36, 2007.
[8]. J. Dean and S. Ghemawat, "Mapreduce: Simplified Data Processing on Large Clusters," Proc. Sixth Symp. Operating System Design and Implementation (OSDI'04), pp. 137-150, 2004.
[9]. E. Dijkstra, "A Note on Two Problems in Connexion with Graphs," NumerischeMathematik, vol. 1, pp. 269-271, 1959.
[10]. E. Cohen, E. Halperin, H. Kaplan, and U. Zwick, "Reachability and Distance Queries via 2-Hop Labels," Proc. 13th Ann. ACMSIAM Symp.Discrete Algorithms (SODA '02), pp. 937-946, 2002.
[11]. D. Hutchinson, A. Maheshwari, and N. Zeh, "An External Memory Data Structure for Shortest Path Queries," Discrete Applied Math., vol. 126, pp. 55-82, no. 1, 2003.
[12]. C. Wang, W. Wang, J. Pei, Y. Zhu, and B. Shi, "Scalable Mining of Large Disk-Based Graph Databases," Proc. 10th ACM Int'l Conf. Knowledge Discovery and Data Mining (SIGKDD '04), pp. 316-325, 2004.
[13]. Goldberg and C. Harrelson, "Computing the Shortest Path: Search Meets Graph Theory," Proc. 16th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '05), pp. 156-165, 2005.
[14]. C. Aggarwal, Y. Xie, and P. Yu, "GConnect: A Connectivity Index for Massive Disk-Resident Graphs," Proc. VLDB Endowment, vol. 2, no. 1, pp. 862-873, 2009.
[15]. C. Mayfield, J. Neville, and S. Prabhakar, "ERACER: a Database Approach for Statistical Inference and Data Cleaning," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIMGMOD '10), pp. 75- 86, 2010.