Scalability and Performance Improvement in Graph Search Query Using FEM Framework

Priyanka S Chumbhale1, Dr. M. U. Kharat2
Student of M.E1, Professor2
Computer Department
MET BKC, Adgaon, Nasik, Savitribai Phule Pune University, Maharashtra, India

Abstract - A shortest path computing can be defined as finding the distance between two nodes in graph and tree based search. In many domains graph data can be used such as, social networks or in knowledge graph. The procedure of graph search consists of sub-graph. The problem to find shortest path between two nodes can be solved using minimal spanning using prim’s or kruskal algorithms, also (TSP) Traveling Salesman Problem algorithm is used. There is problem occurred when given graph is too large to fit in memory, for this reason use of an external memory is come into picture. Previously used Disk-based method suffered from problem when given graph exceeds its size. In our proposed system we are finding the shortest path for efficient relational approaches to graph search queries. For that, our system uses FEM framework. It is used to bridge the gap between relational operations and graph operations. To improve the performance of FEM framework, our system uses a window function and merges statement. Also to improve scalability without indexing we proposed an edge weight aware graph partitioning scheme and design a bi-directional restrictive BFS. As a part of our contribution in this system we provide a generic framework called as FEM-framework. Also we proposed weight aware edge table partitioned methodology. Our proposed FEM framework helps to improve efficiency of proposed system.

Keywords - Graph, Relational database, Shortest path

I. INTRODUCTION

Today, graph based search is widely used for finding shortest distance of given node as there is a rapid growth in uses and popularity of social networks. It is needed when graph search is over the graph. Graph may consist of sub-graph [3]. Sometimes graph is too large to fit in memory then it will exceed the boundary of memory. Previously, disk-based system have limited memory with them therefore, they have some problems as well as some restrictions on graph search query if graph exceeds the main memory or it may use external memory.

Neo4j [2] proposed a technique that supports huge size graph as well as it supports to primitive operations such as, traversal or finding shortest path. This technique has some challenges/difficulties to support general graph search query. We examined that MapReduce framework and its implementation Hadoop [2] have capability of processing huge graph in the distributed system. Relational Database (RDB) is based or it supports graph search. In the study of previous system we noticed that RDB and graph search have some overlapped functionalities, such as, storage, data buffer, index, optimizations, etc. RDB is capable as well as it can maintain complex data types, like, XML data [10], [7]. Our system focused on shortest path search. There are two main reasons behind it, first is shortest path search plays key role in many large applications and second, it have similar evaluation pattern like other search query. Our system improves scalability without indexing. We proposed an edge weight aware graph partitioning scheme as well as designed a bi-directional restrictive BFS. In our system we represented weight aware edge partitioning scheme. Our aim is to improve the performance and scalability without acquiring extra construction and space of the system. To find shortest path between two vertices using recent SQL standard and to improve efficiency of the system we proposed solution called as a FEM framework. It is proposed to work on shortest path discovery. FEM Framework fall the gap between graph operations and relational operations. We use recent SQL standard like window function to merge multiple sql statement in a single query and to improve efficiency of the system.

The scope of our system works on shortest path discovery using relational database queries. This function is not available in all databases like mysql, access, etc. this system can only work with some databases like SQL server, oracle, dB2. This system works only on single graph considering positive weights of the edges. To identify shortest path between two vertices we use SQL queries to identify shortest path. Outsource maximum work to RDB server to find next frontier node. While working with RDB server, use latest and efficient SQL techniques. To improve efficiency of system while searching shortest path, our main contributions in this system are as follows: to provide generic graph FEM-framework for improving efficiency of our proposed system. Also we take an experimental analysis on synthetic and real-life data. In this paper we are going to present, related work in Section II, and then we present the proposed system model, and system architecture in Section III. Next, we give our detailed process of algorithm in Section IV and mathematical model in Section V, then we provide our experimental result in section VI. Finally, we conclude our work in Section VII.

II. RELATED WORK

Jun Gao, Jiashuai Zhou, Jeffrey Xu Yu [1] This system works only on single graph considering positive weights of the edges. To identify shortest path between two vertices we use SQL queries to identify shortest path. Outsource maximum work to RDB server to find next frontier node. While working with RDB server, use latest and efficient SQL techniques. To improve efficiency of system while searching shortest path.
E. W. DrrKrR [2] suggests the solution for forming the tree of minimum length between the n nodes. In this system authors concentrated on solving the problem of searching minimal path between suggested nodes p and Q. In this approach nodes are classified into three sub-categories.

Thomas Stutzle [3] focused on contributions of solving the problem of TSP. This system implements Lin-Kernighan. Authors reviewed some symmetric and asymmetric tsp for utilization of this system. Also heuristic solutions are created using SLS methods. Lin-Kernighan (LK) Algorithm, Population-based ILS Algorithms and The Mimetic Algorithm (MA-MF), ACO Algorithms, Nearest-Neighbor Heuristic (NNH) is proposed for the TSP problem solving.

Michalis Potamias, Francesco Bonchi [4] discussed about the methods that are landmark-based for point-to-point distance estimation in huge networks. It results into ongoing approximate standard Random and the state-of-the-art exact techniques. For the fast estimates of the actual distance in very short time this system uses pre-computed information. This system uses social graphs with explicit or implicit links. This system uses five real-world datasets are used to represent its experimental results. They are also researching about dynamic data structure.

R. Prim [5] represents various algorithms for finding shortest path such as, minimal spanning tree, traveling salesman problem, kruskal algorithm etc. This system aims to maximize the symmetric functions of longest spanning sub-tree of a connected graph. This system identified the problem of about the arithmetizing of metric factors. This system utilized an approach used to manage the quite huge-scale problems. This system checks closed cycles. In this system kruskal described the shortest spanning sub-tree in graph. Traveling salesman problem searches the closed path minimum length.

S. Trißl and U. Leser [6] represents the GRIPP index structure (Graph indexing based on Pre and Post order numbering. It is speedy method for indexing typical and large biological networks. In this reachability queries are important in graph; one can repeatedly traverse the graph at query time, search from start node and then evaluate BFS or DFS search until no more edges remain. In this system graphs are saved as a collection of nodes and edges in an RDBMS. Vertices in the graph contains unique identifier whereas binary relationship between two nodes.

A. Goldberg [7] authors developed bidirectional variants of search and also investigate several variants of the new algorithms. These algorithms are used to compute optimal shortest path for graph. This paper proposed a new lower-building scheme based on landmarks and pre-processing technique for evaluating distance bounds. This approach considered a square grid with integral arc lengths which are selected randomly. Lower-building scheme is important for quality of the bounds. This works on general graphs by taking the advantage of additional information. ALT algorithm is used for best practices. P2P algorithm notices the least vertices in the graph.

B. Bahmani, K. Chakrabarti [8] proposed fast MapReduce algorithm. In this system this algorithm is used to implement PPR approximation algorithm. This system majorly focused on scalability of algorithms. This system does not required new algorithms to researcher about database expert’s machine learning. This approach integrates Weka and database. Basic model used to access the records in the DBMS. Also SQL statements are used for implementation of advanced SQL statements. WekaDB utilizes standard JDBC API.

Beibei Zou [9] proposed a relational database to described as secondary storage that to reduced limitations of previous relational database system. In this system Weka is added as back-tier to relational database system that is known as WekaDB. This system uses Map-Reduce iterations to solve the problem of optimal among a huge family of algorithms. This system Map-Reduce computes single random walks of a given length for all nodes in a graph.

C. Aggarwal, Y. Xie, and P. Yu [10] constructs a connectivity index for massive-disk resident graphs. This system an edge sampling based approach is used to create representations of the underlying graphs. This system studied about the connectivity problem for massive disk-resident graphs. A disk-based query index for connectivity queries is implemented to gain the proposed systems aims. This system formed an index for connectivity queries. This system assumed number of nodes. This system also created an indexed Representation from Compressed Graphs.

C. Wang, W. Wang, J. Pei, Y. Zhu [11] proposed an effective index structure, known as, ADI (for adjacency index). This structure supports different graph patterns over huge databases. For utilization of the system gSpan algorithm is used. This system achieves executable of gSpan scalability. This system examined a sequential order on a representation of all graph patterns to enumerate all frequent graph patterns efficiently. For the formation of data structure of graph ADI index structure is used. A three-level index is used that consist of edges, graph-ids and adjacency information. ADI is flexible structure of storage.

C. Mayfield, J. Neville, and S. Prabhakar [12] proposed an approach which is based on relational dependency networks; it contains an efficient approximate inference algorithm. This system is based on SQL-based, declarative framework. This uses the framework which is able to achieve accuracy comparable to a baseline statistical method using Bayesian networks with exact inference. This system probably works on aspects like, Incomplete and erroneous, Correlated attributes High-level dependencies etc. System proposed a framework that aims to achieve assumes missing values in relational databases is to capture attribute dependencies with graphical models.

David Hutchinson1, Anil Maheshwari, and Norbert Zeh [13] suggests the step by step procedure to save rooted trees in external memory. This system builds a data structure that allows for solutions of shortest path queries using a planar graph. This system solves the problem of shortest path queries of graph. For result of scalability, authors adopt the parallel disk model PDM. It also considered a single processor. PDM contains external memory, which having D disks; D-disks are classified into blocks of B consecutive data items. This system describes a blocking of a rooted tree T therefore user can traverse a bottom-up path from a given vertex v of T in an I/O-efficient manner.

D. Wagner and T. Wilhelm [14] suggested different techniques to speed-up Dijkstra’s algorithm. This system provides guarantee to return a shortest path. It runs
considerably faster. This system overview the developments and extensions of classic results, authors proposed Bidirectional Search performs two types of searches such as Normal search and forward search. A* search, algorithm is used to modify the priority of active nodes to change the order. It modifies the priority, equality.

E. Cohen, E. Halperin [15] introduced simple and natural distance and reachability characterized schemes for directed and undirected graphs. Author proposed an algorithm to build 2-hop hide whose size is greater than the smallest 2-hop hide by a factor of at most O (log n). This system preprocessed a scheme that generates a data structure which is not much greater than the original network. In network, every node of the network allocates the distance label to evaluate the reachability between two nodes. This system considered all undirected graphs, and only worst case results.

III. PROPOSED SYSTEM

The graph dataset i.e. graph vertices and connecting edges with its corresponding weights are saved in database. The whole data is saved in partitioned table. To find shortest path we use Dijkstra’s algorithm and bidirectional BFS algorithm.

As a part of contribution A* algorithm is also included to find shortest path. User can fire query to find path between two vertices. System finds those vertices from table if vertices are from belongs to different tables then bidirectional BFS search is used else Dijkstra’s algorithm is used. SQL queries handle the most time consuming part in search queries. Standard SQL function such as window function and merge statement is used to improve performance of the system. The system is divided in various modules.

FIGURE 1
SYSTEM ARCHITECTURE

Graph generation
The dataset is represented in terms of graph. To define this graph structure we use JAVA FX tool.

User Query Processing
User enters query for shorted path retrieval between 2 nodes. We analyses the graph structure, the RDB structure of graph and identifies whether 2 nodes belongs to the same table or distributed in 2 tables. Depending on the structure we use shortest path identification strategy. Shortest path discovery with FEM is used for single table where as Bidirectional restricted search is used for partitioned table.

Shortest path discovery with FEM
For analysis of shortest path we use Dijkstra’s algorithm. Enlisted in algorithm section, for this algorithm SQL processing is required. The time consuming computations are carried out at RDB end. Minor preprocessing is done at client end. For RDB processing SQL and auxiliary SQL path finding techniques are required. Algorithm section enlists these techniques.

Bidirectional restricted BFS search
Table partitioning strategy is highly used while handling large graph structures. Partitioned tables contain edges between vertices and its weights. A restrictive BFS strategy is used for such partitioned table. In this bidirectional search is started from both the vertices simultaneously. For this SQL in Expansion technique is required. The algorithm and SQL expansion techniques are enlisted in algorithm section.

A* Algorithm
A* algorithm designed to implement the combination of heuristic approaches like BFS and formal approaches like Dijkstra’s algorithm.

Performance Evaluation
For performance evaluation we compare query execution time with respect to methods, databases and operators. Graphical analysis is displayed on the screen.

IV. ALGORITHMS

Algorithm 1: Shortest Path Discovery in FEM framework
Input Set
S as a source node
T as target node and
G for graph i.e. G (V, E)
Where, V is set of vertices in the graph and
E is set of edges in the graph.

Output
The shortest path between source node S and target node T

Process
1) Initialize TA table (i.e. where it stores all visited node) with the source node.
2) Locate the mid value i.e. the ID for next frontier node.
3) Expand the paths.
4) Extract the number of affected tuples and terminate iteration when TA table is not updated.
5) Otherwise finalize the node ‘u’ with its identifier mid.
6) Detect whether the target node has been finalized or not.
7) Once the node is finalized, recover the full path from source to the target node.

Algorithm 2: Bidirectional Restrictive BFS on partitioned tables
Input Set
1. S as a source node
T as target node and
Partitioned tables, TE1, TE2, TEpts. With partitioned vector
Output
A shortest path between source node S and target node T

Process
1) Initialize TA\textsuperscript{f} with node s and TA\textsuperscript{b} with node t.
2) minCost = ∞
3) i=1, j=1 , l\textsubscript{i} =0, l\textsubscript{j} =0, n\textsubscript{f} =1, n\textsubscript{b} =1.
4) While l\textsubscript{i} + l\textsubscript{j} <= minCost do
   if n\textsubscript{i} = n\textsubscript{f} then expand path, n\textsubscript{f} = no. of affected tuples.
   Compute l\textsubscript{i} = i=i+1.
5) Else Perform similar action from line 1 to 4 for backward expansion.
6) Locate minCost.
7) Make verification with visited node table.
8) Compute minimal distance.
9) Locate the node xid in shortest path.
10) Find the sub-path p0 from source node to xid node.
11) Find the sub-path p1 from xid node to destination node.
12) Return p0 + p1

Algorithm 3: A* Algorithm

Input Set
1. Start Node
2. Goal Node

Internal Data
1. Fringe: A list of map locations to be evaluated, in ascending order of estimated distance.
2. Close List: A list of map locations that have been fully evaluated.

Output
Shortest path between Start node and Goal node

Process
1) Firstly Create a search graph G, which consist of the start node, S1. Then insert S1 in OPEN list.
2) Create initially empty CLOSED list.
3) If OPEN list is empty, then exit with failure.
4) Select the first node from OPEN list, remove it from OPEN list, and put it on CLOSED list. Call this node as n.
5) If n is a goal node, then exit effectively with the clarification obtained by tracing a path along the pointers from n to S1 in graph G. (The pointers define a search tree and are recognized in Step 7).
6) Develop node n; generate the set M, of its successors that are not already ancestors of n in graph G. set up these members of M as successors of n in graph G.
7) create a pointer to n from each of those members of M that were not previously in G (i.e. not already on either OPEN list or CLOSED list). Add these members of M to OPEN list. For each member of M that was already on OPEN list or CLOSED list, forward its pointer to n if the best path to m found so far is throughout n. For each member of M already on CLOSED list, forward the pointers of each of its descendants in graph G so that they point backward along the most excellent paths found so far to these descendants.
8) Reorder the OPEN list in increasing order of f values.
9) Go to Step 3

VI. 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
F\textsubscript{1} = Dataset conversion to RDB format
F\textsubscript{2} = Create Data structures TN, TE
F\textsubscript{3} = Generate Graph
F\textsubscript{4} = Analyze query and find type
F\textsubscript{5} = SQL in Path Finding
F\textsubscript{6} = Auxiliary SQLs in Path Finding
F\textsubscript{7} = SQL Expansion
F\textsubscript{8} = Distance Calculation
F\textsubscript{9} = Dijkstra’s algorithm for shortest path discovery
F\textsubscript{10} = Bi-directional Restrictive BFS
F\textsubscript{11} = A* Algorithm
F\textsubscript{12} = Memory Evaluation
F\textsubscript{13} = Time Evaluation
O = \{ O\textsubscript{1}, O\textsubscript{2}, O\textsubscript{3} \}
O\textsubscript{1} = Shortest path
O\textsubscript{2} = Graph visualization
O\textsubscript{3} = Performance Evaluation Report

VII. EXPERIMENTAL RESULTS
We have developed a web application for user to get recommendation and review analysis. A system in build java-JDK1.7. Apache software foundation is used for server side setup. SQL Server 2008 is used to store database. Core i3 machine with 4 GB RAM is used for development and testing. Eclipse IDE is used to test the system.

Dataset details
Live Journal dataset is downloaded from Stanford data collection and it is having community network friendship relation preservation. The evaluation of system is carried out on the basis of memory used and time required for shortest path finding techniques.

Memory Evaluation
A] A* and Dijkstra’s Memory Evaluation
Following is the graphical representation of the memory evaluation of the performance of A* and Dijkstra’s algorithm.
In the above graph red bar represents the A* memory evaluation and blue bar represents the Dijkstra’s memory evaluation. X-axis represents the range of nodes and Y-axis represents memory required for execution in megabytes.

**B) BFS algorithm Memory Evaluation**

Following is the graphical representation of the memory evaluation of the performance of BFS algorithm with partition and without partition.

![FIGURE 3](image3.png)

**BFS ALGORITHM**

In the above graphical representation red bar represents BFS memory evaluation without partitioned and blue bar represents BFS memory evaluation with partitioned. Y-axis represents the memory required in megabytes.

**Time Evaluation**

**A) A* and Dijkstra’s Time Evaluation**

Following is the graphical representation of time evaluation when A* and Dijkstra’s algorithm is in process. These amounts of time in milliseconds are observed for different ranges of records.

![FIGURE 4](image4.png)

**A* AND DIJKSTRA ALGORITHM**

In the above graph red bar represents the A* time evaluation and blue bar represents the Dijkstra’s time evaluation. X-axis represents the range of nodes and Y-axis represents time of execution in milliseconds. X-axis represents the range of nodes.

**B) BFS algorithm Time Evaluation**

Following is the graphical representation of the time evaluation of the performance of BFS algorithm with partition and without partition.

In the graphical representation, red bar represents BFS time evaluation without partitioned and blue bar represents BFS memory evaluation with partitioned and they are placed on X-Axis. Y-axis represents the time required in milliseconds.

**VII. CONCLUSION**

The proposed FEM framework that supports graph search queries and find optimal solution to travel two vertices. RDB is used in shortest path discovery method. Most of the complex operations are transferred to RDB server. For next frontier node search RDB table is used. The table are indexed and hence provide efficiency in search. A SQL query such a window functions and merges function is used to improve performance of the system. The graph table is partitioned and bidirectional restrictive search increases the system efficiency.

**VIII. REFERENCES**


