Computing Shortest Path for Graph Using Fem Framework

Piyush Kulkarni¹, Kapil Vyas²
PG Student¹, Assistant Professor²
Department of Computer Science and Engineering
BM College of Technology, Indore (MP), India

Abstract:
This paper focus on finding the shortest path between two vertices of graph or tree. The graph data emerge in different domain like a social network, web graphs and knowledge graph. The problem of finding the shortest path between two nodes in graph can be solved using prims or kruskal’s algorithm and travelling salesman problem. This problem occurred when graph size increases and the disk-based method has limitations. To finding shortest path our system uses FEM framework which bridge the gap between graph operations and relational operations. We introduced windows function and merge statement to the improve performance of FEM framework. To improve scalability and avoid extra indexing, we proposed an edge weight partitioning schema and bi-directional restrictive BFS.

Keywords: graph indexing, graph search queries, graph, shortest path, database.

I. INTRODUCTION

Today’s world importance of graph increases and graph search faces more challenges. Whenever we use a graph, graph search is more common. Graph search seeks for specific purposes such as minimum travelling tree, salesman travelling path, the shortest path between two nodes. As graph size increase day by day, there is a problem of memory limitation so secondary memory comes into a picture. Neo4j is one who can store large graph into it and provide operation such as shortest path discovery, graph traversal. The performance, stability, and maturity of graph database systems improved continuously, as graph based systems have to implement complex component including storage, query optimization, query evaluation. We notice that MAP Reduced framework and open source Hadoop can process large graph store on distributed file system.

The Relational Database is now stable enough to play a key role in an information system. The graph data management and Relational Database have much similar functionality such as index, storage, optimizations, data buffer. The Relational database also supports specific graph search queries like BFS and reachability queries. However, it requires lots efforts to support generic graph search queries in Relational Database. First, each query has different forms, so implementation of each query is not possible. Second, the semantic mismatch between relational operations and graph operations.

This paper focus on shortest path for two reasons. First, Play a key role in many application like reveal relationship between two individual in a social network. Second, it represents queries which has same evaluation pattern. Our systems increase scalability without indexing. To find the shortest path using SQL standard, we proposed system as FEM framework. WE introduced two new features such as windows function and merge statement. WE also weight aware edge partition schema over restrictive BFS. Our aim is an improve scalability and performance of the system without extra indexing and space.

II. RELATED WORK

B. Bahmani, K. Chakrabarti, and D. Xin [1] Proposed a fast MapReduce algorithm. This algorithm performs a single walk of length starting at each node in graph G. This algorithm is more efficient than other MapReduce algorithm setting. The efficiency of an algorithm depends on many factors like data distribution, computation of job scheduler. The machine time and clock time required for this algorithm is very less.

SilkeTrissl, Ulf Leser [2] Proposed GRRIP index structure. The indexing is important for query performance. We can answer reachability query in a graph using GRRIP index structure. Grip require only linear space and time. GRRIP is implemented as stored procedure in relational database management system and it is easy to integrate into graph application.

S. Srihari, S. Chandrashekar, and S. Parthasarathy [3] Developed SQL based approach for querying large graphs in a relational database management system. They introduced a simple light weight framework in graph application through a tightly-coupled network. This approach increases productivity and performance of a system.

B. Zou, X. Ma, B. Kemme, G. Newton, and D. Precup [4] Proposed relational database as secondary storage in order to eliminate the limitation of main memory as they can handle a limited amount of data. They use machine learning software package, Weka and added a relational database.
storage manager as back-tier to a system. To speed up the execution, some general mining tasks are transfer into a database system.

M. Potamias, F. Bonchi, C. Castillo, and A. Gionis [5] Proposed a system for which uses landmark-based method for point-to-point distance estimation in a large network. The methods selecting a node and marked as landmark and computing offline the distances from each node into a graph to those landmarks. At runtime, the distance in a node is needed, it calculate quickly from previous computation distance.


D. Wagner and T. Willhalm [7] Suggested various techniques to speed up the DIJKSTRA’S algorithm. The author combines various speeds up technique such as goal directional search and bidirectional search. The performance of system increases using speed up technique.

J. Dean and S. Ghemawat [8] Proposed MapReduce programming model. It is use for processing and generating a large dataset. Users specify map function to generate a set of intermediate key/value pair and reduce function merge all intermediate value associated with a same intermediate key. This model mainly used for real world task.

E. Dijkstra [9] Focuses on a problem in connection with graph and its solutions. E Dijkstra gives solutions of problem like find the path of minimum total length between two given node and construct tree of minimum total length between n nodes.

E. Cohen, E. Halperin, H. Kaplan, and U. Zwick [10] Introduced simple and natural distance reachability labeling schemes for directed and undirected graph. They proposed a new data structure for representing all distance in a graph. The data structure is distributed so that label assign to each vertices. Labelling is based on 2 hop cover of the shortest path.

D. Hutchinson, A. Maheshwari N. Zeh [11] Introduced the shortest path queries for planner graph stored on external memory. They stored a rooted tree in external memory so that bottom-up path can be traversed. They represent I/O efficient algorithm for triangular planner graph.

C. Wang, W. Wang, J. Pei, Y. Zhu, and B. Shi [12] Developed index structure to support mining various graph pattern over a large database that cannot fit into main memory. The index structure can be easily adopted by various graph patterns mining algorithm, for an example gSpan algorithm.


III. PROPOSED SYSTEM

The graph is nothing but vertices connected by edges and stored on a database system. The system architecture mainly works on Diikstra’s algorithm and bi-directional BFS algorithm. We also use A* algorithm to find the shortest path between two nodes. The system architecture shows consist of query processing, type identification, shortest path using FEM framework and bi-directional restrictive BFS, estimate time calculation for query processing, graph generation, and report generation. The user fire query to find the shortest path. We also use SQL feature like windows function and merge statement.

Query Processing:

In query processing consist of two new feature such as windows function and merge statement. The user enters a query for finding the shortest path in a graph between two nodes. In this part of the system, we determine a strategy of find the shortest path as two nodes whose path need to be find. If both node in same table we use FEM framework otherwise Bidirectional restrictive BFS.
**FEM Framework:**
FEM framework mainly works on Dijkstra’s algorithm. The FEM framework works in three steps. First, select a frontier node from a visited node. Second, start expansion from the frontier node. Third, merge expanded a node in a visited node.

**Bidirectional Restrictive BFS**
Bidirectional restrictive BFS allow expansion in both a direction that is forward and backward directional. In bi-directional restrictive BFS, multiple node select as frontier node and expansion start from this node in set-at-time fashion and visited node re-expanded. to handle large graph, partitioned table strategy is used.

**Graph generation:**
This shows a complete graph from a source node to destination node. JAVAFX tool is used for graph generation.

II. ALGORITHMS
There are on three different algorithms.

1) Dijkstra’s algorithm for shortest path discovery
2) Bidirectional restrictive BFS on partitioned tables
3) A* algorithm with FEM.

**Algorithm 1: Dijkstra’s algorithm for shortest path discover in FEM framework**
Input: Source node s, target node t graph G (V, E)
Where V is vertices and E is edges in a graph.
Output: shortest path between s and t
1) Initialize table TA with all source node
2) locate mid and ID for next frontier node i.
3) Expand path in forward and backward direction using mid
4) if no tuple affected then break
5) Finalize the frontier node i with mid
6) Repeat step 2 to 5 till node exist
7) Find path p from parent node
8) Return p;

**Algorithm 2: Bidirectional restrictive BFS on partitioned tables**
Input: source node s target node t partitioned table TE1, TE2,…,TEn with partitioned vector
Output: shortest path between s and t

**Algorithm 3: A*algorithm with FEM**
Input: source node s, Target node t, open list, close list
Output: shortest path between s and t
1) Initialize TA^f with source node s and TA^b with target node t
2) MinDist $\leftarrow \infty$;
3) $i \leftarrow 1$, $j \leftarrow 1$ where i= number expansion forward direction j= number of expansion in backward direction
4) $l^f_i \leftarrow 0$, $l^b_j \leftarrow 0$ where $l^f_i$ maximal finalize distance in forward expansion $l^b_j$ maximal finalize distance in backward direction
5) $n^f_i \leftarrow 1$, $n^b_j \leftarrow 1$ $n^f_i$ and $n^b_j$ is total number of frontier node in forward and backward direction
6) While $l^f_i + l^b_j \leq$ MinDist do step 7 to 12
7) if $n^f_i \leq n^b_j$ then do step 8to 10
8) Expand path where $n^f_i$ = number of affected tuples
9) Compute maximal finalize distance $l^f_i$
10) $i=i+1$
11) repeat step 7 to 10 in backward expansion
12) locate MidDist from TA^f and TA^b
13) Make verification with table TA^f, TA^b and TE
14) Compute minimal distance
15) Locate node mid in shortest path
16) Find shortest path p1 from s to mid
17) Find shortest path p2 from mid to t
18) Return shortest path $p = p_1+p_2$
5) If it is on list then fire Select command for selecting the first node i.e frontier node from visited node on OPEN list as remove it from OPEN list, and put it on CLOSED list and expand the frontier node. Call this node as n1. If n1 is target node then exit by tracing the path along with pointer from n1 to s in graph G.

6) Develop node n1, generate set S1 of successor of node

7) Create pointer to n1 from each member of set M

8) Add a member of set M to open list.

9) For each member of set m in the open list or close list, the forward pointer to n1 to found path from n1.

10) For each member on a close list a forward pointer in each node in descendant in graphs G to find the path.

11) Expand the next node from frontier node. Merge the expanded node and visited node.

12) Reorder open list in increasing order.

13) Go to step 4

V. MATHEMATICAL MODEL

Let,

\[ S = \text{System}, \]
\[ I = \text{Input set of the system}, \]
\[ O = \text{Output of system}, \]
\[ P = \text{Set of processes of the system}, \]
\[ R = \text{Rule set applied on the system during processing} \]

Therefore, representation of system in set of form is

\[ S = \{ I, P, R, O \} \]

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 = User Query

I2 = Graph

O = \{O1, O2, O3\}

O1 = Shortest path between two node

O2 = Performance Evaluation Report

O3 = Graph visualization

F = \{F1, F2, F3, F4, F5, F6, F7, F8, F9, F10\}

F1 = Analyze query and identify type

F2 = Merge statement

F3 = windows function

F4 = SQL Expansion

F5 = Dijkstra’s algorithm for shortest path discovery

F6 = A * algorithm of path finding

F7 = Create Data structures TN, TE

F8 = Bi-directional Restrictive BFS

F9 = Generate Graph

F10 = Distance Calculation

VI. EXPERIMENTAL RESULTS

All methods of the system implemented in java on NetBeans IDE. The system method evaluate on Apache software foundation server. To store data, we use SQL Server 2008 database. The Core i3 processor with 4 GB RAM machine is used for deployment and testing. The system consists of login window, control panel, data search and report generation. The user login into the system using username and password. The control panel consists of various options like data search, report generation, and Logout.
The main output window shows the shortest path between two nodes and complete graph as well.

The results consist of time evaluation with and without a partition in BFS and A* algorithm and Dijkstra’s algorithm. The window shows the performance results of each algorithm.

The red bar and blue bar represents time evaluation without partition and with partition respectively. The performance of with partition is better than without partition.

From experimental result, we reach to the following conclusions:
- We can say that FEM framework improves the performance and scalability of graph.
- A* algorithm with FEM speed up the process of finding shortest path.
- The new SQL feature such as window function and merge statement improve the performance of FEM framework.
- We optimize basic method via the bi-directional restrictive BFS an overweight aware partitioned edge table which can improve performance and scalability without extra overhead.
VIII. REFERENCES


