A Novel Approach of Graph DB for Distributed Computing Environment

Akash Patel
Department of Information Technology Engineering
SVIT, Vasad, Anand, Gujrat, India

Bhavesh Patel
Department of Information Technology Engineering
SVIT, Vasad, Anand, Gujrat, India

Abstract- Graph database is a very new paradigm of database that will show new horizons to the world for storing and retrieving structured data. In this paper we propose a system that will work on basic of graph theory and helps to manage the graph database well. This technique will efficiently distributes the graph database and decompose the graph for very efficient parallel query processing. This model has advantage of reducing read wrote latency over the conventional graph database systems.

Keywords – Graph Theory, Graph database, Distributed Computing.

I. INTRODUCTION

We are using relational database since 1970s. We are producing data at so much high rate that relational database can't handle data well. No SQL database is revolution in database industry nowadays. Social media is very active nowadays lots of pictures, videos, comments, likes are overlaying whole world. Youtube is popular for video and video size tends to be very large. It says that every 1 minute users upload 60 hrs of video on youtube. According to website worlds 90% data is generated in last two years. Smartphone are come first in producing the data for every individual. Now a day's everyone has smart phones and producing enormous amount of data in terms of lists, photos, mails, attachments, apps download, emails etc. It is very difficult to manage and process these data generated by smart phones. Moreover other things are also there which is generating the data rapidly.

Graph databases deal with one of the great industry trends of today: leveraging compound and vivacious relationships in highly connected data to generate insight and passive advantage. Whether we want to understand relationships between customers and dealers, elements in a telephone or data center network between clients, entertainment producers and consumers, or genes and proteins, the ability to understand and analyze vast graphs of highly attached data will be key in decisive which companies better their competitor over the coming decade. Although large companies realized this some time ago and began creating their own proprietary graph processing technologies, we're now in an era where that technology has rapidly become popular.

Today, general-purpose graph databases are a reality, enabling regular users to experience the reimbursement of connected data without having to invest in build their own graph road and rail network. A graph is just a group of vertices and edges—or, a set of nodes and the relationships that join them. Graphs represent entities as nodes and the ways in which those entities relate to the world as relationships. This general-purpose, easy-to-read composition allows us to model all kinds of set-up, from the building of a rocket ship, to a system of infrastructure and from the supply-chain or origin of groceries, to medical history for populations.

II. PROPOSED SYSTEM

A. Multi level graph partitioning techniques –

This technique reduces magnitude of the graph by merging vertices together. Then it computes partition on this reduces graph, And finally projects this partition on original graph.

There are three steps:

(1)First phase reduce magnitude of graph by merging vertices. Merging of vertices does by iterative process. By this process a coarser graph is created. This is done until a certain small graph is achieved.

(2)Second phase a partitions graph with the smallest magnitude – the coarsest graph – is computed.

(3)In the third and last phase, the computed partition is iteratively projected back to the original graph.

In each of these steps heuristic method are applied. Objective of heuristic method is to produce a solution in a reasonable time frame that is good enough for solving a problem. K-way partitioning technique divides vertex set into k smaller partitions. Because of this characteristic Metis can partition a graph into equal graph while cutting down the cut edges to minimum.

This is the reason we chose Metis as our partitioner tool. Now second is graph database so we selected Neo4j as our graph database. Let's look at internal structure of neo4j.

B. Neo4j System Structure –

The Neo4j stores all records of graph data as disk based files. Neo4j uses two layers of cache file system cache and object cache. Relationship contains most of the information on disk. A nodes relationship classified according to relationship type and its good to traverse rapidly. Neo4j guaranties the ACID property for the transactions. Top layer is API layer. It provides functionality for business related applications. Core APIs are shared by all instance of Neo4j.

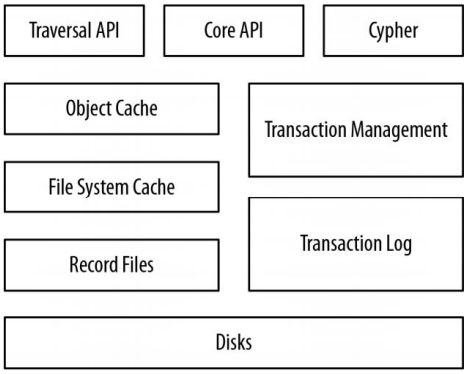

Figure 2: Logical Structure of Neo4

Nodes: The node data stored in nodestore.db file. Each node is 9 bytes in length. First byte is in-use flag which tells whether a record is in use or not. Next four bytes are of first relationship connected to the node. Last four bytes the id of the first property of the nod.
Relationship: File stored as neostore.relationstore.db on disk. Relationship has fixed size 33 byres. Each record contains id of starting node and ending node.
Property: Properties stored in a file name neostore.propertystore.db as well it contains property index store file neostore.propertystore.db.index.

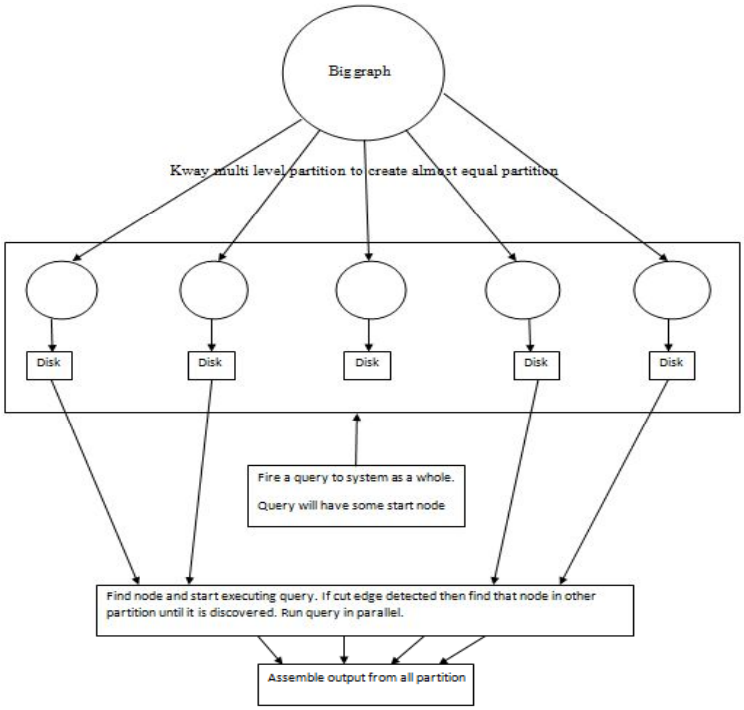

Figure 3: Proposed system

In our proposed system there is a big graph data, this graph data will then be divided into small partitions through K-way multilevel partitioning technique to create almost equal partition of graph data. Then we will store each partitioned graph into different disks meaning on different machine. Each machine preloaded with neo4j setup. Now user will see this whole setup as one system because this is not known to user. Now user will query system as whole. The first node (start node) will be specified in the query and as query begin to execute it will find cut edges and will propagate through all partitions in which the cut edge is found and thus query running in parallel through all partition and the performance will increase.

This is based on graph theory. Our approach will save the read-write latencies. Graph database of current generation uses the distributed storage solutions for distribution of graphs or they replicate data for better availability. This approach is convenient for small storage system of clusters which have small amount of memories. Plus the added advantage is query is running parallel and the output is assembled in the end. This will increase the overall performance of the database. For this experiment we are taking dataset from Stanford university website.

REFERENCES

[1] Mark Graves, Ellen R. Bergeman, and Charles B.Lawrence, "Graph Database systems" *IEEE ENGINEERING IN MEDICINE AND BIOLOGY December 1995,737-745.*
[2] Andrew Lumsdaine and Douglas Gregore Bruce Hendricson and Jonathan Berry,"CHALLENGES IN PARALLEL GRAPH PROCESSING" *Parallel Processing Letters World Scientific Publishing Company January 2007.*
[3] Renzo Angles, "A Comparison of Current Graph Database Models" *IEEE 28th International Conference on Data Engineering Workshops December 2012, 171 – 177.*
[4] Hongcheng Huang, Ziyu Dong, "Research on architecture and query performance based on distributed graph database Neo4j" *IEEE DEC 2013, 533 – 536.*
[5] Rik Van Bruggen "Graph Database for Enterprise Architects" *Neo Technology, September 2014.*
[6] George Karypis and Vipin Kumar, "A fast and high quality multilevel scheme for partitioning irregular graph", *Society for Industrial and Applied Mathematics, 1998, 359-392.*
[7] http://mashable.com/2012/06/22/data-created-every-minute/
[8] http://nosql-database.org/

[9] http://db-engines.com/en/article/Key-value+Stores
[10] http://ravendb.net/docs/article-page/2.0/csharp/intro/what-is-a-document-database
[11] http://www.mongodb.com/document-databases
[12] https://www.cs.rutgers.edu/~pxk/417/notes/content/bigtable.html
[13] http://whatis.techtarget.com/definition/graph-database
[14] http://www.neo4j.org
[15] http://thinkaurelius.github.io/titan/
[16] http://glaros.dtc.umn.edu/gkhome/views/metis
[17] http://markorodriguez.com/2011/02/08/property-graph-algorithms/
[18] https://snap.stanford.edu/data/
[19] Graph Database by O'REILLY media Written by: Ian Robinson, Jim Webber, Emil Eifrem.