Distributed Knowledge Graph Querying on Edge, Fog and Cloud

Shriram R.

Department of Computational and Data Sciences
Indian Institute of Science, Bangalore 560012 INDIA
Email: shriramr@iisc.ac.in

***Abstract*—Knowledge graphs with millions of vertices and edges are currently being used to enable smart devices and knowledge discovery. IoT (Internet of Things) devices are becoming the consumer and producer of such large graphs. Existing graph querying and processing systems are designed to run on commodity hardware. However, IoT applications need lightweight graph processing systems that can scale across large number of devices and handle concurrent queries with low latency. In this project, a lightweight distributed graph querying engine is proposed which can natively scale across edge, fog and cloud devices. It supports declarative query model for vertex, edge, reachability, shortest path and subgraph pattern mining queries. The engine has a novel caching, indexing and query partitioning mechanism which enables to scale efficiently with graph size and query workload. The engine is evaluated on a large scale virtual IoT testbed with massive real-world knowledge graphs. The results indicate that the engine can provide x% lower latency on a diverse query workload and can handle y% more queries concurrently compared to the baseline system.**

I. Problem Description

The problem that is being solved in this project is the scaling of graph querying on edge, fog and cloud devices which can provide low-latency query respone for many concurrent queries on massive knowledge graphs. *Figure 1* shows a representation of typical edge-fog-cloud layout. There is a hierarchical structure where fogs can communicate with edges and cloud and the edges can directly communicate with only fog. Each edge is associated with a single fog depending on its spatial proximity or any other parameter. The bandwidth and latency of edge-fog, fog-fog and fog-cloud network links are significantly different from each other.

The edge, fog and cloud devices are heterogenous and have different compute and storage capabilities. In addition, the device failure rate varies across the layers with edge having the highest failure rate. In this project however, all the devices are considered to be 100% reliable.

Existing graph processing/querying systems are designed for commodity hardware and cannot be efficiently deployed on low powered edge devices like smartphones, smart home appliances etc. These systems do not take into account the network hierarchy and device heterogeneity

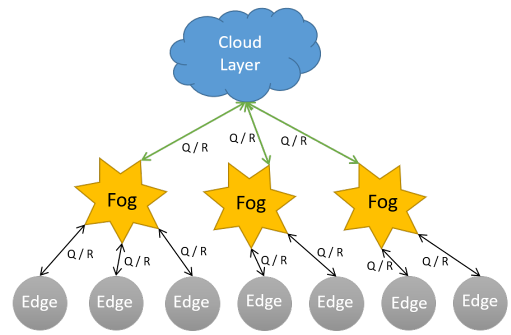

Fig. 1: Edge-Fog-Cloud Layout

that exists on the current edge-fog-cloud layout. The proposed graph querying engine uses GoDB which is a distributed graph database as the backend and implements the novel features on top of the database.

In this project, following are the key contributions:

- **Graph Partitioning:** A novel graph partitioning approach is used which partitions the graph based on the locality of query. This means that the portion of knowledge graph which is required by a query is identified and moved closer to the device performing the query automatically
- **Query Planning:** A query planner which can efficiently partition the query predicates according to graph locality and placement thereby optimizing query cost is implemented
- **Query Caching:** A novel query caching system is devised which caches recent queries by edges and their corresponding results in the fog layer. It exploits the temporal and spatial locality of queries to provide low latency respone. It also has a novel cache replacement policy and can match an incoming query with cache.
- **Graph Indexing** Index data structures from existing literature that can accelerate the processing of queries are built in a distributed fashion and are used for pruning the knowledge graph for processing. Different type of indexes are built for each query type and are maintained dynamically as per updates to the knowledge graph

II. Novelty/Scalability

This problem is novel and has the following scalability challenges: The size of real-world knowledge graph data is massive with respect to the storage available in edges. For example, YAGO which is a semantic knowledge base has about 10 million nodes and 120 million edges along with various properties. It has a total size of 170 GB (uncompressed).

Many of the graph mining problems and pattern matching (subgraph isomorphism) are NP complete which means that they cannot be feasibly computed in real time using just the edge devices.

The knowledge graphs are relatively dynamic in nature with multiple updates throughout the graph happening in a short time interval. This poses a challenge in terms of building and updating indexes rapidly.

Graph partitioning is non-trivial due to the structure and heterogeneity of edge-fog-cloud. Many of the IoT applications require real-time answers whereas the existing distributed graph engines typically address batch processing queries with high latency.

An edge-fog-cloud system deployed on a city-wide network can have on the order of thousands of devices. The query engine has to efficiently scale to this size and simultaneously provide acceptable level of performance.

Existing query engines are built on platforms or frameworks that are resoure heavy. In order to run the proposed engine on the edge device, a significant optimization of the system in terms of resource requirements is neccessary.

III. Related Work

Pregel, *Giraph* and *GraphX* are designed as batch mode graph processing frameworks for commodity clusters which are not suitable for interactive queries.

Trinity is a distributed graph engine built as a inmemory key-value store. Though it is possible to perform interactive graph querying, the engine depends on high-speed network and requires large memory both of which are not feasible in edge-fog-cloud.

GC focusses on caching graph queries and results for graph isomorphism problems. It also provides novel cache replacement strategies for different workloads. However, it does not address other query types and is not designed for hierarchical device layout like in our case.

C-Tree and *Views* provide efficient indexing techniques for subgraph isomorphism queries. However, these indexes are not designed for distributed systems where the index has to be partitioned along with the graph data.

FERRARI describes an efficient index for reachability and path queries. However, it is not straightforward to translate the design for our distributed setup.

GraphS provides an efficient indexing technique to perform cycle detection in large distributed dynamic graphs with low latency. Our engine can adopt this technique for heterogenous device setup.

GoDB is one of the closest work to the proposd engine. It is a distributed graph database which supports declarative queries on large property graphs. *GoDB* is built on *GoFFish* subgraph-centric batch processing platform thereby using its scalability.

GoDB supports vertex, edge, path and reachability queries. It also come with a novel cost model which uses execution heuristics to come with a optimim query plan that minimizes latency.

However, it is not designed to work on heteregenous device layout and is not capable of running on the edge devices. It comes with indexing mechanism for filtering vertices and edges based on properties but lack query type specific indexes which can dramatically reduce the query processing time.

GoDB also do not have any caching mechanism and therefore cannot leverage on temporal and spatial locality of input queries.

Quegel system provides a vertex centric programming model for processing graph on commodity clusters. It treats queries as first class citizens and uses a novel superstep-sharing execution model along with *Hub* indexing technique to improve utilization of resources and performance.

However, it lacks a declarative query model and graph partitioning technique to support heterogenous devices and targets only commodity clusters.

It seems apparant that there are no graph querying system that solves the eaxct problem of graph querying in edge-fog-cloud.

IV. Problem Definition and Approach

The proposed graph consists of different modules spanning across the edge, fog and cloud layer.

V. Proposed Experiments

[1] [1].

References

[1] Y. Simmhan, S. Aman, A. Kumbhare, R. Liu, S. Stevens, Q. Zhou, and V. Prasanna, "Cloud-based software platform for big data analytics in smart grids," *Computing in Science Engineering (CiSE)*, vol. 15, no. 4, pp. 38–47, 2013.

[1]Please refer to the IEEE Taxonomy for more details. http://www.computer.org/portal/web/publications/acmtaxonomy