An Empirical Study on Recent Graph Database Systems

Ran Wang[1], Zhengyi Yang[2], Wenjie Zhang[2], and Xuemin Lin[2]

[1] East China Normal University, Shanghai, China
rwang@stu.ecnu.edu.cn
[2] The University of New South Wales, Sydney, Australia
{zyang,zhangw,lxue}@cse.unsw.edu.au

Abstract. Graphs are widely used to model the intricate relationships among objects in a wide range of applications. The advance in graph data has brought significant value to artificial intelligence technologies. Recently, a number of graph database systems have been developed. In this paper, we present a comprehensive overview and empirical investigation on existing property graph database systems such as Neo4j, AgensGraph, TigerGraph and LightGraph[3]. These systems support declarative graph query languages. Our empirical studies are conducted in a single-machine environment against on the LDBC social network benchmark, consisting of three different large-scale datasets and a set of benchmark queries. This is the first empirical study to compare these graph database systems by evaluating data bulk importing and processing simple and complex queries. Experimental results provide insightful observations of various graph data systems and indicate that AgensGraph works well on SQL based workload and simple update queries, TigerGraph is powerful on complex business intelligence queries, Neo4j is user-friendly and suitable for small queries, and LightGraph is a more balanced product achieving good performance on different queries. The related code, scripts and data of this paper are available online[4].

Keywords: Graph database systems · Labeled property graph · LDBC benchmark.

1 Introduction

Graphs are widely used to model the intricate relationships among entities in real world. The recent blossoming of modern graph technologies has brought tremendous value to artificial intelligence (AI) and machine learning (ML) applications [25]. The related context in graph data enhances AI in many ways, ranging from graph based acceleration of ML, connected feature extraction, explainable AI, and especially, to the area of knowledge engineering. For example, using context-rich knowledge graphs for recommendation at eBay [27], extracting

[3] LightGraph has recently renamed to TuGraph.
[4] https://github.com/UNSW-database/GraphDB-Benchmark

knowledge from the "Lessons Learned" database at NASA [23], and combining graph features and ML for fraud detection in security and finance [12].

Graph management systems, as one of the most fundamental infrastructures in graph-based AI and ML applications, have received lots of attention both in industry and academia. Two types of graph management systems are developed to manage graph data efficiently, effectively collecting explicit and implicit information. The first one is *graph analytics systems*, performing batch computations on large clusters, such as GraphLab [22], Giraph [2] and GraphX [16], etc. The latter is *graph databases*, fulfilling primitive requirements of storing and fast querying graph data, such as Neo4j [8], JanusGraph [5] and ArangoDB [3], etc. Specifically, graph databases primarily adopt labeled property graph model or resource description framework (RDF). As RDF databases have received lots of attention in literature [11] and the recent explosive popularity of property graph databases [9] such as Neo4j, in this paper, we aim to comprehensively evaluate graph databases that support labeled property graph model. Note that our focus in this paper is on the four popular enterprise graph database systems Neo4j, AgensGraph [1], TigerGraph [10] and LightGraph [6], though there are also graph databases proposed in the literature [17, 18].

A number of graph databases have been used in enterprises to handle fast-evolving graph data in AI domains. Even though some popular graph databases have already taken a large share of the market and also been studied in academic communities, in recent years, a few new graph database systems, such as AgensGraph, TigerGraph and LightGraph, are developed to satisfy the needs of higher level business demands, lower resource consumption, and enhanced performance. Inspired by these observations, in this paper, we empirically investigate major enterprise graph databases, and take the most popular Neo4j and several young graph databases AgensGraph, TigerGraph, LightGraph as examples, to evaluate their performance in detail under various experimental settings.

In the literature, various benchmarks [15, 21, 26] have been proposed to evaluate the performance of graph database systems. Additionally, social network is penetrating into every aspect of daily lives and marketing activities, producing enormous information. Thus, inspired by and following existing work [24], we employ Linked Data Benchmark Council Social Network Benchmark(LDBC_SNB) [15, 26] as our evaluation tool, under which our experiments evaluate four graph databases by loading three different scale datasets and processing interactive and business intelligence query workloads.

The main contributions of this paper are as follows:

- We investigate the market of recent enterprise graph database systems, and present a study of prevalent products. Due to the rapid advance in graph databases, existing works [21, 24] are either outdated or incomplete.
- Take four products Neo4j, AgensGraph, TigerGraph, and LightGraph as examples, we make further investigations on graph databases. To the best of our knowledge, this is the first work comprehensively survey and compare popular choices of modern industrial-strength graph database systems.
- We adapt all queries in the LDBC_SNB benchmark and make significant adjustments to work with all four aforementioned graph databases. Based

on this, we systematically evaluate them including the performance of data loading and querying under three different data scales.
- We thoughtfully analyse and conclude the results of our unified benchmark. This provides researchers and companies with insightful advice on how to select a proper graph database system in different use cases.

The rest of this paper is organized as follows. Section 2 states necessary preliminaries. Section 3 reviews related works. Section 4 presents our investigation. Section 5 shows experimental results. Finally, we conclude the paper and discuss future research directions in Section 6 .

2 Preliminaries

2.1 Labeled Property Graph Model

A labeled property graph (LPG) is a directed graph with *labels* and *properties*, in which labels define different subsets (or classes) of vertices and edges and properties are arbitrary number of key-value pairs representing real-world attributes. The LPG model can easily represent many complex relations in real applications and has the advantages of high expressiveness. In addition, it is nature to understand making it increasingly popular in the database community.

2.2 LDBC Benchmark

LDBC_SNB [15] models a social network akin to Facebook, consisting of persons and their activities. It is rich in entities, relationships and attribute types, simulating the characteristics of information architecture in a real-world social network. It proposes three query workloads: *Interactive*, *Business Intelligence*, and *Graph Algorithms*. Since the last one is applicable for graph analytics systems which is out of the scope of this paper, we only take the first two as our testing objects, covering the aspects of evaluating the ability of commercial graph databases in processing actual business requirements.

Interactive workload [15] defines user-centric transactional-like interactive queries, consisting of 8 transactional update(IU), 7 simple read-only(IS) and 14 complex read-only(IC) queries. Among them, IU queries are simple vertex or edge insertions. Both of IS and IC queries are information retrieval operations, starting with a specific vertex. The difference is that IS targets at simple pattern matching, accessing at most 2-hop vertices and collecting basic data, while IC defines complex path traversal and returns computed and aggregated results. Business intelligence(BI) workload [26] defines 25 analytic queries to respond to business-critical questions. Different from interactive workload, BI queries start with multiple vertices and contains more complex graph analysis and result statistics operations, aiming at collecting valuable business information.

The structure of LDBC_SNB dataset and detailed definitions of workloads can be found in the official specification [20].

3 Related Work

With the development of graph databases, many evaluation and benchmark endeavors are conducted to compare these products. For existing evaluation works, some [13, 19] only focus on listing the characteristics of graph databases without any empirical exploration, some [14, 21] restrict evaluation on micro operations, small scale data and a limited set of products, and some [24] aim at present the superiority of their own products. Meanwhile, many existing benchmarks cannot evaluate the ability of graph databases in processing actual and business requirements. For instance, micro-benchmark [21] is limited in micro operations, and RDF-benchmark [11] is only suitable for finding RDF structures.

The experimental study in our paper fills the gap. We not only conclude the characteristics of prevalent and young enterprise graph databases, but also evaluate the performance of representative products on large-scale datasets, employing benchmark LDBC_SNB which simulates a realistic social network and defines both of micro and macro query workloads.

4 Graph Database

Section 4.1 reviews the features of enterprise graph database systems. Then, a detailed introduction of evaluated databases is presented in Section 4.2.

4.1 An Overview of Graph Databases

Graph database is a type of NoSQL database emerged as a response to the limitations of traditional relational databases. Graph databases are highly optimised for handling connected data. Popularity of graph databases dramatically increases in recent years because of their high performance and the ease of use. We give an overview of graph database systems in the current commercial market, including well-developed graph and multi-model database systems, such as Neo4j, JanusGraph (successor of Titan) and ArangoDB, and other relatively young systems, such as AgensGraph, TigerGraph, LightGraph and Nebula [7]. Their basic features are concluded in Table 1, including database type, storage structure, open source or not(Op.), supporting distributed processing or not(Ds.), transactional or not(Ta.), schema-free or not(Sf.), implementation languages(Impl.) and query languages(Lang.).

From Table 1, we can see that, as enterprise products, these graph database systems are almost all transactional with ACID guarantees. Moreover, most native graph databases prefer self-designed storage structures and graph query languages, while hybrid databases are more flexible and support multiple storage engines or query languages. Besides, with the increasing requirements for processing large data, many products target at high scalability, supporting distributed storage of data and can be deployed in a cluster of machines. Among all studied graph databases, only TigerGraph and LightGraph are not open source projects, so the studies of their storage technologies are limited.

Table 1. Basic information of graph database systems.

System	Type	Storage	Op.	Ds.	Ta.	Sf.	Impl.	Lang.
Neo4j	Native	Linked Lists	Yes	No	Yes	Yes	Java	Cypher
JanusGraph	Hybrid	Cassandra/HBase	Yes	Yes	Yes[5]	No	Java	Gremlin
ArangoDB	Hybrid	MMFiles/RocksDB	Yes	Yes	Yes	Yes	C++	AQL
AgensGraph	Hybrid	PostgreSQL	Yes	No	Yes	Yes	C	Cypher,SQL
TigerGraph	Native	Native Engine	No	Yes	Yes	No	C++	GSQL
LightGraph	Native	Native Engine	No	No	Yes	No	C++	Cypher
Nebula	Native	RocksDB	Yes	Yes	No[6]	No	C++	nGQL

Then we choose several systems to conduct further research and performance evaluation based on the benchmark LDBC_SNB. To summary, the selection criteria are: 1) support the labeled property graph model, 2) support declarative graph query languages, 3) support OLTP, 4) could fully implement query workloads in LDBC_SNB, and 5) full licence available. Therefore, we take Neo4j, AgensGraph, TigerGraph and LightGraph to investigate and benchmark. Especially the last three, as young graph databases, get little attention in existing work even though they demonstrate promising performance in their own reports.

4.2 Graph Databases Details

Neo4j. Neo4j [8] is the most popular graph database system. Compared with other products, mature community is one of its biggest advantages. It provides user-friendly web interface and APIs and supports many third-party applications, frameworks and programming language drivers. Neo4j also developed the graph query language Cypher, which is a widely-used and easy-reading language. However, Neo4j typically cannot scale to large graphs as it dose not support data sharding in distributed environment. Even in the enterprise version, the distributed mode just replicate the data in all machines to achieve high availability. With the storage structure of linked lists, Neo4j stores vertices, edges and attributes natively and separatively. This design usually causes high memory consumption and low efficiency, even if indexes are created [24].

AgensGraph. AgensGraph [1] is proposed as a new generation multi-model graph database. As a multi-model database, AgensGraph supports graph, relational, document and key-value data models at the same time, adopting SQL and Cypher as query languages. Thus, it can integrate the two languages in one single query, supporting more flexible use cases and very friendly to experienced SQL-users. AgensGraph adopts the PostgreSQL RDMS as storage engine. However, when processing large data, it will take a very long time to load and consume large memory to store data. In the process of querying complex requirements, a large temporary space will be occupied. AgensGraph is a developing product and cannot support all grammars in Cypher. In this work, we implement queries with high complexity in LDBC_SNB by integrating Cypher and SQL.

[5] The transaction isolation levels in JanusGraph is determined by the choice of storage.
[6] The transactional support in Nebula is still under development.

TigerGraph. TigerGraph [10] is one of the rising stars in distributed graph database in recent years. Its core system is developed from scratch using C++. By combining native graph storage with MapReduce, massively parallel processing and fast data compression/decompression, TigerGraph shows strong scalability and great performance, especially in handling complex queries. More importantly, TigerGraph can be easily deployed to a large scale of clusters and process queries distributedly, allowing it to answer queries on extremely large graphs that are likely to fail in a single machine setting. It also develops its own query language GSQL, which is a powerful and procedure-like language. However, as a non open source commercial product, TigerGraph is not freely available.

LightGraph. LightGraph [6] is a high-performance graph database product, implemented in C++. As an enterprise-focused product, LightGraph is still under development, but has already demonstrated impressive strength. LightGraph supports storing and querying billion-scale data in single machine. Meanwhile, it adopts a lockless design, greatly improving the throughput under high loads and enabling queries to be processed with high parallelism. Although LightGraph provides Cypher interface, it still cannot support most grammars. Therefore, for complex querying requirements, it suggests to implement stored procedures by Python or C++ APIs. Please note that, we only implement IU and IS micro queries with Cypher, and for IC and BI queries in the benchmark, we directly use the plug-in offered by the authors to run.

Other Databases. JanusGraph [5], previously known as Titan, is a massively scalable graph database and natively integrates with the Apache TinkerPop framework. As a hybrid system, it can integrate with third-party systems like HBase, Cassandra, ElasticSearch for storing and indexing the graph structures. ArangoDB [3] is a native multi-model database, supporting graph, document and key-value models, and provides a unified database query language to coverer three models in a single query. Nebula [7] is a newly released high-performance and scalable graph database capable of hosting very large scale graphs with low latency, getting more attention recently.

However, JanusGraph adopts an imperative query language Gremlin[7]. Meanwhile, both query languages in Nebula and ArangoDB are still developing and not fully compatible with all LDBC_SNB query workloads at this moment. Since only Cypher and GSQL are in the process towards an upcoming international standard language for property graph querying [4], we only consider Cypher- and GSQL-based systems, that is Neo4j, AgensGraph, TigerGraph and LightGraph.

5 Empirical Studies

Section 5.1 presents the experimental setup. Then, we discuss experimental results in Section 5.2. After that, an overall evaluation is given in Section 5.3.

[7] Gremlin has limited degree of declarative support.

5.1 Setup

We generated three datasets with different scales by using LDBC data generator [15]. All datasets have the same structure [20]. The statistics of datasets are summarized in Table 2. We can find that the number of vertices and edges and the raw size of datasets increase almost linearly with scale factor.

Table 2. Statistics of datasets.

Dataset	Scale Factor	$\|V\|$ (Million)	$\|E\|$ (Million)	Size(GB)
DG1	1	3.182	17.256	0.798
DG10	10	29.988	176.623	8.257
DG100	100	282.638	1775.514	85.238

For benchmark LDBC_SNB, there are overall 54 queries in interactive and business intelligence workloads. Their implementations in four graph databases are acquired from their official staff or implemented by ourselves, like BI queries in AgensGraph, IU and IS queries in LightGraph, guaranteeing the optimality of query statements in each database as much as possible. To conduct a fair comparison, we take the same parameters for the same query in all systems. Note that, for LightGraph, as its Cypher interface has not been fully implemented and optimized, we only implemented IU and IS queries with Cypher, and used C++ stored procedures provided by the authors to do other experiments. The other three databases were operated by query languages.

We conducted all experiments on a machine with two 20-core processors Intel Xeon E5-2680 v2 2.80GHz, 96GB main memory, and 960G NVMe SSD, runnig Ubuntu 16.04.5 operating system. We tested performance on fast NVMe SSD storage, as it is already a default option for database systems. To perform experiments, we ran each micro query (IU and IS) 100 times and each macro query (IC and BI) 3 times, setting mean running time as result. By default, we set timeout as 1 hour for each query processing, and use symbols 'TO' and 'OOM' to represent timeout and out of memory respectively.

5.2 Experimental Results

Data Importing. Due to the large size of raw data, datasets were bulk imported into graph database systems. Data importing is also a common operation in production. We present the data importing time of four databases and the storage size of each loaded dataset respectively in Fig. 1.

Figure 1-a shows that, for all three scale datasets, the data importing time in Neo4j is the shortest, followed by LightGraph, TigerGraph and AgensGraph. Actually, even if we consider the indexes creating time in Neo4j and preprocessing time of stored procedures in LightGraph, both of them still present better bulk importing performance than TigerGraph and AgensGraph. AgensGraph, limited by its third-party relational storage engine PostgreSQL, the speed of data bulk importing is much slower than other systems. For DG100, AgensGraph will take about one day to load data, unreasonable in common usage.

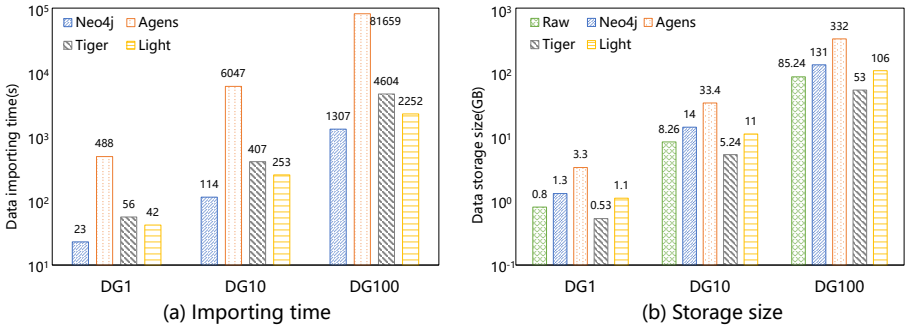

Fig. 1. The bulk importing time and storage size of datasets.

Figure 1-b presents the raw and loaded storage sizes for three datasets in four databases, showing that TigerGraph requires the least space to store data, even less than raw data, followed by LightGraph, Neo4j and AgensGraph. The storage consumption in LightGraph is about twice as much as that in TigerGraph, Neo4j is about three times large. AgensGraph requires the largest storage space.

Based on the above experiments, Neo4j performs best in the efficiency of large data importing, but its storage cost is very large. TigerGraph costs least space to store data, but take a little long time to load data. LightGraph presents good performance overall. AgensGraph shows the worst performance.

Processing Interactive Queries. Interactive workload consists of IU, IS and IC queries. Among them, both of IU and IS are micro operations. IC are macro queries under more complex interactive scenarios. We select several representative results to present and full data are available online. Since the running time of IU and IS operations is not more than one second, we record them in milliseconds, and record the running time of IC queries in seconds.

Table 3. Running time(millisecond) for IU queries.

IU	DG1				DG10				DG100			
	Neo4j	Agens	Tiger	Light	Neo4j	Agens	Tiger	Light	Neo4j	Agens	Tiger	Light
2	4.62	**0.46**	8.14	0.56	4.45	**0.52**	9.81	0.63	4.13	**0.54**	10.36	0.98
4	30.50	8.47	8.55	**1.02**	40.56	7.94	8.52	**1.07**	38.41	9.75	8.90	**1.07**
6	5.02	2.08	8.38	**1.39**	16.10	3.53	8.99	**1.58**	12.01	2.99	9.72	**1.52**
8	1.14	**0.42**	8.18	0.69	1.13	**0.50**	8.45	0.56	1.28	**0.50**	9.40	0.61

Table 4. Running time(millisecond) for IS queries.

IS	DG1				DG10				DG100			
	Neo4j	Agens	Tiger	Light	Neo4j	Agens	Tiger	Light	Neo4j	Agens	Tiger	Light
1	1.05	1.83	3.47	**0.60**	1.45	1.70	3.27	**0.61**	1.60	1.85	3.65	**0.66**
2	18.49	10.01	10.90	**8.90**	33.11	21.40	**11.09**	15.96	25.80	26.20	**9.91**	16.00
4	1.33	**0.33**	4.01	0.53	1.11	**0.34**	3.74	2.34	0.57	**0.34**	4.06	0.61
6	1.33	13.11	4.66	**0.67**	2.37	14.61	4.7	**0.65**	1.34	15.40	4.41	**0.70**

Table 3 lists the processing time of IU queries, showing that LightGraph and AgensGraph are more efficient than other two databases. LightGraph is good at inserting vertices, such as IU_4 and IU_6, and AgensGraph is good at inserting edges, like IU_2 and IU_8. Neo4j is faster than TigerGraph in updating edges, while slower in updating vertices.

The experimental results of IS queries are listed in Table 4. We find that LightGraph is the most efficient in most cases. TigerGraph and AgensGraph only show good performance in several cases, like IS_2 and IS_4, and worse than Neo4j in most cases according to the full data.

Table 5. Running time(second) for IC queries.

IC	DG1				DG10				DG100			
	Neo4j	Agens	Tiger	Light	Neo4j	Agens	Tiger	Light	Neo4j	Agens	Tiger	Light
1	0.60	0.32	**0.03**	0.06	2.36	1.17	**0.12**	0.36	10.22	8.26	**0.56**	2.48
3	2.01	0.35	0.06	**0.01**	24.06	7.95	0.37	**0.05**	616.54	63.82	1.32	**0.37**
6	2.58	0.17	0.09	**0.01**	113.92	0.36	0.31	**0.03**	TO	5.66	0.97	**0.11**
10	0.50	0.71	**0.03**	0.04	2.32	2.93	**0.06**	0.12	9.33	12.41	**0.15**	0.37
12	0.19	2.58	**0.02**	0.04	0.66	4.96	**0.06**	0.15	0.60	55.97	**0.13**	0.16
13	0.01	**0.01**	0.01	0.01	0.03	0.02	**0.01**	0.01	**0.00**	0.03	0.02	0.01
14	340.55	43.34	0.20	**0.01**	424.05	488.61	0.27	**0.02**	63.83	TO	0.31	**0.03**

Table 5 shows the running time of IC queries, there are big differences in performance between graph databases. Overall, both of TigerGraph and LightGraph present efficient performance with little difference, while AgensGraph and Neo4j perform much worse than other two systems. In the case of executing IC_14 for DG1 and DG10, Neo4j is even four orders of magnitude slower than LightGraph. However, for IC_13, which requires to return the length of the shortest path between two given vertices, the results show little difference among all databases across all datasets. This is because both Neo4j and AgensGraph support the keyword `shortestPath` and optimize internal computing process.

For interactive workload, all databases work well in micro queries, but show big differences in macro queries. LightGraph performs best in most cases. TigerGraph is only efficient in processing complex queries, while bad at micro operations. AgensGraph and Neo4j are suitable for micro operations, wile work much worse than other two databases when processing IC queries.

Processing Business Intelligence Queries. Business intelligence workload, considering more actual business applications, requires higher performance for databases to response to complex scenarios. Table 6 lists 12 representative queries as examples, recording results in seconds.

Actually, all graph databases cannot successfully execute all BI queries across all datasets. Overall, TigerGraph performs best, followed by LightGraph. Neo4j and AgensGraph still perform bad, the results are timeout in many cases. In the case of executing BI_7 for DG1, Neo4j is four orders of magnitude and AgensGraph is five orders of magnitude slower than LightGraph, showing big performance difference. Although there may be some reasons in query statements

Table 6. Running time(second) for BI queries .

BI	DG1				DG10				DG100			
	Neo4j	Agens	Tiger	Light	Neo4j	Agens	Tiger	Light	Neo4j	Agens	Tiger	Light
2	3.36	6.67	0.59	**0.44**	29.16	102.2	**5.27**	9.19	237.6	1388.2	**42.67**	221.8
4	1.58	0.21	**0.02**	0.03	14.50	0.62	0.12	**0.09**	173.3	4.76	**1.19**	3.57
7	375.8	1647.8	0.88	**0.04**	TO	TO	0.88	**0.74**	TO	TO	OOM	**38.63**
8	0.41	1.65	**0.03**	0.08	3.89	6.46	**0.16**	1.00	43.31	69.90	**1.32**	60.07
10	941.2	48.15	0.05	**0.03**	TO	692.2	**0.31**	0.40	TO	TO	**4.11**	33.91
13	0.77	1.39	0.18	**0.12**	5.56	14.36	1.63	**1.07**	46.35	415.5	**13.21**	82.66
16	3.36	TO	**0.49**	0.58	39.97	TO	**4.59**	6.62	429.9	TO	**50.64**	426.1
17	0.41	0.27	0.01	**0.01**	34.65	3.55	**0.03**	0.06	TO	999.1	**0.20**	0.88
18	6.44	352.6	**0.37**	2.95	76.00	TO	**4.71**	58.02	700.1	TO	**52.57**	1742.2
20	4.73	36.53	1.21	**0.84**	44.53	255.1	**14.69**	30.94	732.7	TO	OOM	**290.8**
23	0.18	0.24	**0.02**	0.05	1.55	1.20	**0.06**	0.37	13.44	11.22	**0.51**	50.10
24	16.25	29.96	0.78	**0.66**	191.5	373.2	**7.57**	14.73	TO	TO	**75.71**	1198.7

optimization, the biggest problem lies in the technologies of data storing and query processing. For example, for BI_16, AgensGraph is timeout even under the DG1. By splitting its original query statement, we found that AgensGraph cannot match pattern `(Person)-[:KNOWS*3..5]-(Person)` in a reasonable time, which maybe related to its storage structure or query execution mechanism.

In the comparison of TigerGraph and LightGraph, for DG1, their performances show little difference. But with the increasing scale of datasets, the processing time in LightGraph increases faster than that in TigerGraph. Thus, TigerGraph is much more efficient under DG10 and DG100, in part, thanks to is build-in parallel processing mechanism and stored procedure-like query statements implemented by GSQL. However, for BI_7 and BI_20 in DG100, TigerGraph is out of memory, but LightGraph finishes them successfully, showing that TigerGraph requires more memory when processing large datasets.

5.3 Overall Evaluation

We only list part of experimental results, full data is available online. According to full results, the four databases present different performance and no one can perform best in all scenarios. The findings can be summarized as follows:

- Neo4j is user-friendly and the most efficient one in data importing. However, it is only suitable for micro queries and small-scale datasets, and shows bad performance in running complex business intelligence queries.
- AgensGraph works well on SQL accompanied workload and simple update and query operations, while performs very bad in processing complex queries and managing large datasets.
- TigerGraph is powerful on complex business intelligence queries, such as IC and BI queries, especially on large datasets like DG100. It need the least memory to store data, although takes a little long time to load data.
- LightGraph is a more balanced product, achieving good performance on all types of queries. However, it cannot fully support Cypher grammars, and complex queries only can be implemented by stored procedures.

Although we cannot get a closer look at the implementation details of closed source products TigerGraph and LightGraph, we list three potential reasons of their performance advantages over community-driven graph databases, namely, Neo4j and AgensGraph. The first reason is the language differences in implementation as TigerGraph and LightGraph are implemented in C++, which generally shows a greater performance than the language implements Neo4j, namely, Java. The second reason is the other two systems are schema-free, which TigerGraph and LightGraph both have fixed schema, which allows more optimizations to be done. The third reason is commercial products tend to use advanced algorithms and optimizations to improve their efficiency. Finally, as for the hybrid system, AgensGraph, the extra layer to its underlaying relational database encounters significant extra costs to the graph database overall.

These four graph database systems are all business products, stable, professional and suitable for processing actual and business query requirements. According to our experience in usage, Neo4j and AgensGraph are more user-friendly, supporting more complete query language grammars. The query statements of TigerGraph and LightGraph are more like stored procedures, requiring some technologies in expressing queries. Most importantly, the two products are not free available, users cannot change or optimize by themselves.

6 Conclusion and Future Work

In this paper, we investigate and conclude some graph database systems. Moreover, we present a further research and evaluation on Neo4j, AgensGraph, TigerGraph, LightGraph. Based on the benchmark LDBC_SNB and single machine environment, experiments across three different scale datasets show that LightGraph and TigerGraph have significantly better performance in managing large data and processing high complexity queries. Neo4j and AgensGraph are suitable for simple micro operations and give friendly use experience. In the future work, we will extend our study to more graph database products, and evaluate their performance under distributed environment.

Acknowledgments Xuemin Lin is supported by 2019B1515120048, 2018AAA-0102502 and 2018YFB1003504.

References

1. Agensgraph. `https://bitnine.net/`
2. Apache giraph. `http://giraph.apache.org`
3. Arangodb. `https://www.arangodb.com/`
4. Gql. `https://www.gqlstandards.org/`
5. Janusgraph. `https://janusgraph.org/`
6. Lightgraph. `https://fma-ai.cn/`
7. Nebula. `https://nebula-graph.io/cn/`
8. Neo4j. `https://neo4j.com/`
9. Ranking of graph dbms. `https://db-engines.com/en/ranking/graph+dbms`

10. Tigergraph. https://www.tigergraph.com/
11. Abdelaziz, I., Harbi, R., Khayyat, Z., Kalnis, P.: A survey and experimental comparison of distributed sparql engines for very large rdf data. Proceedings of the VLDB Endowment **10**(13), 2049–2060 (2017)
12. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data mining and knowledge discovery **29**(3), 626–688 (2015)
13. Besta, M., Peter, E., Gerstenberger, R., Fischer, M., Podstawski, M., Barthels, C., Alonso, G., Hoefler, T.: Demystifying graph databases: Analysis and taxonomy of data organization, system designs, and graph queries. arXiv preprint arXiv:1910.09017 (2019)
14. Ding, P., Cheng, Y., Lu, W., Huang, H., Du, X.: Which category is better: Benchmarking the rdbmss and gdbmss. In: Proceedings of the 2019 APWeb & WAIM. pp. 207–215. Springer (2019)
15. Erling, O., Averbuch, A., Larriba-Pey, J., Chafi, H., Gubichev, A., Prat, A., Pham, M.D., Boncz, P.: The ldbc social network benchmark: Interactive workload. In: Proceedings of the 2015 ACM SIGMOD. pp. 619–630 (2015)
16. Gonzalez, J.E., Xin, R.S., Dave, A., Crankshaw, D., Franklin, M.J., Stoica, I.: Graphx: graph processing in a distributed dataflow framework. In: Proceedings of OSDI'14. pp. 599–613 (2014)
17. Hao, K., Yang, Z., Lai, L., Lai, Z., Jin, X., Lin, X.: Patmat: A distributed pattern matching engine with cypher. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. pp. 2921–2924 (2019)
18. Kankanamge, C., Sahu, S., Mhedbhi, A., Chen, J., Salihoglu, S.: Graphflow: An active graph database. In: Proceedings of SIGMOD'17. pp. 1695–1698 (2017)
19. Kolomičenko, V., Svoboda, M., Mlỳnková, I.H.: Experimental comparison of graph databases. In: Proceedings of 2013 iiWAS. pp. 115–124 (2013)
20. LDBC SNB task force: The ldbc social network benchmark. Tech. rep., LDBC (2019), https://ldbc.github.io/ldbc_snb_docs/ldbc-snb-specification.pdf
21. Lissandrini, M., Brugnara, M., Velegrakis, Y.: Beyond macrobenchmarks: microbenchmark-based graph database evaluation. Proceedings of the VLDB Endowment **12**(4), 390–403 (2018)
22. Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., Hellerstein, J.M.: Distributed graphlab: A framework for machine learning and data mining in the cloud. Proceedings of the VLDB Endowment **5**(8), 716—-727 (2012)
23. Meza, D.: How nasa finds critical data through a knowledge graph. https://neo4j.com/blog/nasa-critical-data-knowledge-graph/
24. Rusu, F., Huang, Z.: In-depth benchmarking of graph database systems with the linked data benchmark council (ldbc) social network benchmark (snb). arXiv preprint arXiv:1907.07405 (2019)
25. Sahu, S., Mhedhbi, A., Salihoglu, S., Lin, J., Özsu, M.T.: The ubiquity of large graphs and surprising challenges of graph processing. Proceedings of the VLDB Endowment **11**(4), 420–431 (2017)
26. Szárnyas, G., Prat-Pérez, A., Averbuch, A., Marton, J., Paradies, M., Kaufmann, M., Erling, O., Boncz, P., Haprian, V., Antal, J.B.: An early look at the ldbc social network benchmark's business intelligence workload. In: Proceedings of the 1st ACM SIGMOD Joint Workshop on GRADES-NDA. pp. 1–11 (2018)
27. Wang, X., Wang, D., Xu, C., He, X., Cao, Y., Chua, T.S.: Explainable reasoning over knowledge graphs for recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 5329–5336 (2019)