图数据库的实证研究

An Empirical Study on Recent Graph Database Systems

fzb0316@163.com

1 摘要

本文使用LDBC-SNB（LDBC social network benchmark）对四个图数据库 （Neo4j, AgensGraph, TigerGraph ,LightGraph(现在改名为TuGraph)）进行比较。本文是第一次研究结果：单机环境下评估数据批量导入、简单及复杂查询。

AgensGraph可以很好地处理基于SQL的工作负载和简单的更新查询，TigerGraph对复杂的商业智能查询功能强大，Neo4j用户友好，适合小型查询，而LightGraph是一个更平衡的产品，在不同的查询上实现了良好的性能。

2 Introduction

现在图数据对ML和AI的好处：三个例子（eBay使用上下文知识图进行推荐、NASA使用Lessons Learned database提取知识、结合ML和图feature进行安全和金融诈骗的检测）

2.1 两种图管理系统：

- 图分析系统：在大集群中进行batch计算（GraphLab、Giraph、GraphX ）
- 图数据库：满足存储和快速查询的基本要求（ Neo4j、JanusGraph、ArangoDB，主要采用带标签的属性图和资源描述框架RDF）

2.2 主要工作和贡献：

- 使用LDBC_SNB作为评估工具，通过加载三个不同规模的数据集并处理交互和商业查询workload来评估四个图数据库
- 调查最近商业图数据库的市场，并介绍流行产品的研究
- 评估上述四个图数据库，本文是首次全面调查和比较现代商业图数据库系统的文章
- 对不同场景适合的图数据库提出建议

3 相关知识和相关工作

3.1 带标签的属性图（LPG）

LPG是具有标签和属性的有向图，其中标签定义了顶点和边的不同子集（或类），属性是表示真实属性的任意数量的键值对。

LPG模型在实际应用中可以很容易地表示许多复杂的关系，并且具有高表达性的优点。此外，由于它方便理解的特性，在业界越来越流行。

3.2 LDBC基准

LDBC-SNB模拟了一个类似Facebook的社交网络，由个人及其活动组成。它具有丰富的实体、关系和属性类型，模拟了现实社会网络中信息架构的特征。它提供了三种查询工作负载：交互式、商业智能和图算法。图算法适用于图分析系统，超出了本文的范围。本文仅将前两个作为测试对象，涵盖评估商业图数据库处理实际业务需求的能力。

交互workload定义了以用户为中心的事务交互查询，包括8个事务更新（IU）、7个简单只读查询（IS）、14个复杂只读（IC）查询。IU是简单的顶点或边的插入，IS和IC查询都是从特定顶点开始的信息检索操作，不同的是，IS的目标是简单的模式匹配，最多访问2跳顶点并收集基本数据，而IC定义复杂的路径遍历并返回计算和聚合的结果。

商业智能（BI）查询开始于多个顶点并包含复杂的图关系和结果统计的操作，旨在收集有价值的业务信息。

关于LDBC-SNB的详细介绍，官方文档说的很详细：LDBC-SNB Specification

3.3 相关工作

随着图数据库的发展，researchers进行了许多评估和基准测试来比较这些产品。对于现有的评估工作，一些只关注于列出图数据库的特征，而没有进行任何实证探索；一些限制于对微观操作、小规模数据和有限的产品集的评估；还有一些旨在展示自己产品的优势。同时，许多现有的基准测试无法评估图数据库处理实际和业务需求的能力。例如，微基准仅限于微操作，RDF基准仅适用于查找RDF结构。

本文的实验研究填补了这一空白。不仅总结了流行的和新的企业图数据库的特点，而且还使用LDBC SNB模拟了现实世界的社交网络，并定义微观和宏观查询工作负载，评估了代表性产品在大规模数据集上的性能。

4 图数据库

4.1 概述

一种针对传统关系数据库的局限性而出现的NoSQL数据库。

表一总结了现在图数据库的基本特征，包括数据类型、存储结构，是否开源（Op.）、是否支持分布式处理（Ds.）、是否支持事务处理（Ta.）、是否无模式（Sf.）、实现语言（Impl.）和查询语言（Lang.）

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[a]	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[b]	No	C++	nGQL

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

从表中可以看到，所有的图数据库几乎都是事务性的，具有ACID保证。此外，大多原生数据库喜欢自行设计存储结构和查询语言，而混合数据库更灵活，支持多种存储引擎和查询语言。

随着处理大数据的需求的不断增加，许多图数据库以高可扩展性为目标，支持数据的分布式存储、并且可以部署在机器集群中。其中TigerGraph和LightGraph不开源。

- **选择Neo4j, AgensGraph, TigerGraph ,LightGraph的原因：**

1） 支持标签属性图模型，2）支持声明性图形查询语言，3）支持OLTP，4）可以在LDBC SNB中完全实现查询工作负载，5）可用的完全许可证。

4.2 图数据库的详细介绍

Neo4j：最流行的图数据库系统，成熟的社区是其最大的优势之一。提供用户友好的web界面和API，并支持许多第三方应用程序、框架和编程语言驱动程序。开发了图查询语言Cypher，一种广泛使用且易于阅读的语言。**但是**，Neo4j不能扩展到大型图，不支持分布式环境中的数据共享。即使在企业版中，分布式模式也只是复制所有机器中的数据。通过链表的存储结构，以本地和单独的方式存储顶点、边和属性。即使创建了索引，这种设计也通常导致改内存消耗和低效率。这篇论文对比了Neo4j与TigerGraph，也研究了其存储[参考论文]。

AgensGraph：仍在开发中。作为一个多模型数据库，同时支持图、关系、文档和KV数据类型，可以在一个查询中集成两种语言。**但是，**在处理大数据时，加载并消耗大量内存来存储数据，这需要很长时间；在复杂查询中，会占用大量的临时空间，不能支持Cypher的所有语法。本文通过集成Cypher和SQL在LDBC SNB中实现复杂查询。

TigerGraph：结合了本地图存储与MapReduce、大规模并行处理和快速数据压缩/解压缩，显示出强大的可扩展性和出色的性能。**重要的是，**可以轻易地部署到大规模集群中，分布式处理查询，还可以解决在单机中可能失败的超大图上的查询（没明白什么是单机中可能失败的查询）。就是不免费，不开源。

LightGraph：仍在开发中。支持在单机中存储和查询十亿级数据，同时，采用无锁设计，提高了高负载下的吞吐量，并使查询能够高并行度处理。**但是，**虽然提供了Cypher接口，但它无法支持大多数语法，对于复杂的查询建议使用Python或C++ API实现存储过程。本文只使用Cypher实现IU和IS微查询，对于IC和BI直接使用作者提供的插件来运行。

其他：JanusGraph(原名Titan)，是一个可大规模扩展的图数据库，与Apache TinkerPop框架本地集成，可以与Hbase等第三方系统集成。**ArangoDB**是一个原生的多模型数据库，支持图、文档和kv模型，并提供了一种统一的数据库查询语言，可以在单个查询中覆盖三个模型。**Nebula**是一个最新发布的高性能可扩展的图数据库，能够以低延迟托管超大规模的图。

5 实验

5.1 实验环境

使用LDBC数据生成器生成了三个不同规模的数据集。所有的数据集具有相同的结构。

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

从表2可以发现，顶点和边的数量以及数据集的原始大小几乎与比例因子呈线性增长。对于LDBC SNB，交互和商业智能workload总共有54个查询，本文尽可能保证每个数据库中查询语句的最佳性，对所有系统中的同一查询使用相同的参数。**注意，**由于LightGraph的Cypher接口尚未完全实现和优化，因

此只使用Cypher实现IU和IS查询，并使用作者提供的C++存储过程进行其他操作。其他三个数据库由查询语言操作。

在一台搭载Ubuntu 16.04.5操作系统、20核处理器Intel Xeon E5-2680 v2 2.80 GHz，96GB主内存和960G NVMe SSD的机器上进行了所有的实验。测试了NVMe SSD存储的性能，因为它已经是数据库系统的默认选项。

实验将每个微查询（IU和IS）运行100次，每个宏查询（IC和BI）运行3次，将平均运行时间设置为结果。若查询时间超过1小时就会被标记为超时，用符号"TO"、"OOM"分别表示超时和内存不足。

5.2 实验结果

5.2.1 数据导入

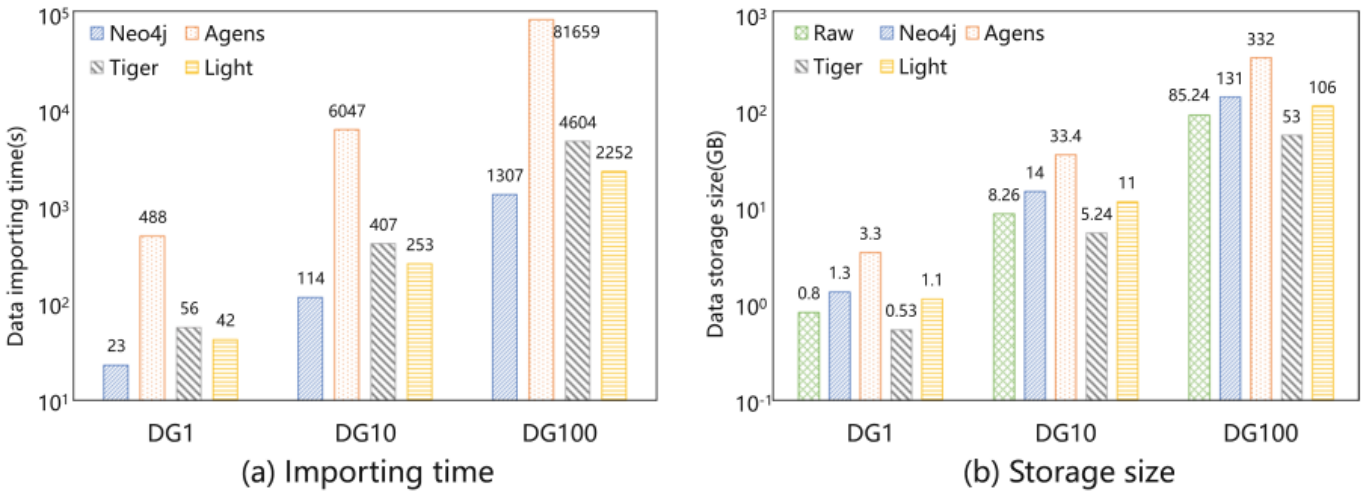

Fig. 1. The bulk importing time and storage size of datasets.

图1-a显示，对于三个规模的数据集，Neo4j的数据导入时间最短。即使考虑到Neo4j的索引创建时间和LightGraph中存储过程的预处理时间，它们仍然比另外两个具有更好的批量导入性能。而AgensGraph受第三方存储引擎PostgreSQL的限制，它的数据导入速度比其他系统慢很多。**注意，**AgensGraph导入DG100的时间大约一天。

图1-b表示四个数据库中三个数据集的原始和加载存储的大小。图中表明，TigerGraph存储数据所用的空间最少，比原始数据还少；

原文：基于以上实验，Neo4j在大数据导入效率方面表现最好，但其存储成本非常大。TigerGraph存储数据所需的空间最少，但加载数据所需时间稍长。LightGraph总体表现良好。AgensGraph显示最差的性能。

5.2.2 处理交互式查询（IU、IS、IC）

IU和IS是微操作，IC是宏操作

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

表3表明，对于IU操作，LightGraph和AgensGraph比其他两个数据库更高效。LightGraph擅长插入顶点（IU4和IU6），而AgensGraph擅长插入边（IU2和IU8）。Neo4j更新边的速度比TigerGeaph快，更新定点的速度慢。对于Operation的序号代表什么类型的数据库操作，在这个文档有详细的表述LDBC-SNB Specification，IU在4.5.1节

- IU2：给定两点（不同类型）的ID，在两点之间添加一条边
- IU4：添加一个点，随之带来两个类型的边
- IU6：添加一个点，随之带来四个类型的边
- IU8：在两个Presons之间添加一条knows边

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**

表4，对于IS查询，大多数情况下LightGraph是最有效的，而TigerGraph和AgensGraph仅在一些情况下（IS2和IS4）性能略好，大多数情况下性能低于Neo4j。操作描述在LDBC SNB文档的5.2。

- IS1：给定一个Person节点ID，检索其属性（姓、名、生日、IP地址、浏览器）和与其相连节点City的名字
- IS2：给定一个Person节点ID，检索其发表的最后10个message。对每个message（评论），返回该message的原始帖子以及其帖子的作者；如果该message是原始帖子，那么会返回两次
- IS4：给定一个message的ID，检索其内容和创建日期

- IS6：给定一个message的ID，检索包含该message的Forum以及该论坛的主持人。由于评论（我认为是message的一种）是不直接包含在论坛中的，则若该message是评论，返回其回复的评论所包含的原始帖子的论坛

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**

表5表明，对于复杂查询，四个图数据库之间的差距很大。总的来看，TigerGraph和LightGraph的性能最好，而Neo4j与AgensGraph比其它两个差得多（DG1和DG10执行IC14时，甚至差四个数量级）。IC13是返回两个给定顶点之间的最短路径长度，所有数据库基本没有差异，这是因为Neo4j和AgensGraph都优化了内部计算，也支持给定关键字的最短路径搜索。

综上，所有数据库在微查询性能相当，但在宏查询（复杂查询）中表现了很大的差异。大多数情况下，LightGraph性能最佳，TigerGraph只在IC中性能较好，而AgensGraph和Neo4j只适用于微查询，IC情况下差很多。

- IC1：给定起点Person的ID，检索以Knows[1..3]连接的Person，返回检索到person的名字，其公司、学校和与原始Person的路径长度（1..3）
- IC3：给定起始Person的ID，检索给定的两个国家（X,Y）以外、在日期（data，data+duration）内发表帖子/评论的与起始Person的关系Know[1..2]（即朋友和朋友的朋友）的Person。
- IC6：给定起始Person的ID和一个Tag（帖子的标签）名字，检索在与起始Person关系Knows[1..2]内Person发表的帖子中，与给定Tag一起出现的其他Tag的名字
- IC10：给定起始Person的ID，查找其朋友的朋友（限制：生于给定月份的21号当天或者之后，并且出生于下个月22号之前），计算每个查找到的Person与原始Person的相似性。相似性定义：
    - common=好友创建的帖子数（限制：该帖子起始Person感兴趣的标签）
    - uncommon=好友创建的帖子数（限制：该帖子不包含起始Person感兴趣的标签）
    - 相似性=common−uncommon
- IC12：给定一个起始Person，查找其朋友对帖子的评论（评论限制：直接评论，帖子限制：包含给

定标签类或该类子类中的标签的帖子），统计评论的数量，并收集该评论原始帖子的标签，返回至少有一个评论、评论计数和标签的Person

- IC13：给定两个Person的ID，在以Knows*0..关系的子图中，搜索这两个Person的最短路径，返回路径的长度：-1表示没有路径；0表示给的是同一个人；>0表示找到了路径
- IC14：基于交互子图，在两个给定ID的Person之间找一个cheapest路径。交互子图是基于Person-Knows-Person子图的，还包括两个Person之间有帖子和评论的交互。edge weight=max[40-args(交互数量)，1]

5.2.3 处理商业智能查询

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

表6显示，总体来看，TigerGraph表现最好，其次是LightGraph。Neo4j和AgensGraph的性能仍然很差，很多情况下会超时（BI7，DG1，Neo4j差了四个数量级，AgensGraph差了5个数量级），虽然跟查询语句优化可能有原因，但最大的问题还是数据存储与查询处理技术的差异。对于BI16，AgensGraph超时，但作者通过查分其原始查询语句，发现其无法在合理的时间内匹配模式（(Person)-[:KNOWS*3..5]-(Person)），这可能与查询执行机制有关。

比较TigerGraph与LightGraph，对于DG1，他们性能差别不大，但随着数据集规模的增加，TigerGraph的性能明显更好。部分原因是它内置了并行处理机制和GSQL实现的类似存储过程的查询语句。然而，对于BI7和BI20，TigerGraph内存不足，这表明TigerGraph需要更大的内存（可能是以空间换时间）。

- BI2：在给定的日期开始的100天时间窗口中，查找使用了给定标签类中标签的message，并与随后

的100天时间窗口进行比较。计算两个窗口的消息数量

- BI4：按国家查找最受欢迎的论坛，论坛的受欢迎程度是根据论坛在该国家中成员数量来决定的，论坛需要在给定的日期之后建立的。计算前100个最受欢迎的论坛，如果一个论坛在多个国家中都受欢迎，那么他应该只计算一次成员最多的国家。如果出现平局，选择ID较小的论坛。返回：在这100个论坛的每个成员中，统计他们在其中任何一个论坛中发的message数量，没发过message的记为0
- BI7：查找具有给定标签的message，检索这些message的直接评论中的标签（不包含原始标签），按名称给这些标签分组，并返回每个组中的评论数
- BI8：给定一个标签，查找所有对这个标签感兴趣的人或者写了包含该标签的message的人（message创建的日期需要在给定日期之后）。对每个人，计算分数：100（如果感兴趣，不感兴趣是0）+num（写的包含该标签message的数量）；此外，还计算该Person的朋友的分数总和
- B10：给定一个Person的ID，检索与其以Knows关系连接的在给定国家的所有Person，这些Person与原始Person以最短路径连接到一个子图中。对这个子图的每个节点，检索至少包含一个属于给定标签类的标签的所有message，对每个message，检索其所有标签。返回：按Person和标签分组，计算该人写的包含该标签的message数
- BI13：在给定国家中寻找zombies（在给定日期之前创建的Person，并且在创建日期到给定日期之间，每月创建一条message）并返回zombies分数。分数=从其他zombies收到赞的数量/收到赞的数量（分母是0，分数记为0，赞只考虑在创建日期到给定日期之间获得的）
- BI16：给定两个标签/日期对（tagA/dataA，tagB/DataB）,对每对（tagX/tagY），在Person之间创建一个Knows子图，对其中每对Person，都在第一个dataX的下一天创建了包含tagX的message，只考虑子图中给定hop之内的一对Person，对这些人，用tagX计算在dataX上创建的message数量，但会至少一条message的Person，按message数量降序。
- ……

5.3 实验总结

上文只列出了部分数据，详细数据在这：数据（需要挂代理）。

结论：

- **Neo4j是用户友好的（有很好的可视化），也是数据导入中最高效的。然而，它仅适用于微查询和小规模数据集，在运行复杂的商业智能查询时表现出很差的性能，而且不支持分布式**
- **AgensGraph可以很好地处理SQL伴随的工作负载以及简单的更新和查询操作，但在处理复杂查询和管理大规模数据集方面表现很差**
- **TigerGraph在复杂查询和商业智能查询（IC和BI）上功能强大，特别是在大型数据集（如DG100）上。它存储数据所需的内存最少，但加载数据所需时间稍长**
- **LightGraph是一款更加平衡的产品，在所有类型的查询中都能取得良好的性能。然而，它不能完全**

支持Cypher语法，复杂的查询只能由存储过程实现

原因：（没有TigerGraph和LightGraph的源码，只有Neo4j和AgensGraph的社区版）

1. **语言差异。TigerGraph和LightGraph是使用C++实现，比Neo4j（JAVA）有更高的性能**
2. **Neo4j和AGensGraph是无模式的。TIgerGraph和LightGraph有固定的模式，运行进行更多的执行优化**
3. **商业的产品更倾向于使用先进的算法和优化来提高效率**
4. **AGensGraph是混合系统，其底层的PostgreSQL会给图数据库带来额外的成本**

最后，虽然TigerGraph和LightGraph很好，其查询语句更像是在直接操作存储，但是不开源，我们没有办法对系统进行更改和优化

fzb0316@163.com