CHALMERS

Viability of Graph Databases in Pipeline Traceability Systems

Bachelor's thesis in Computer Science and Engineering

David Andersson
Max Villing

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

DEGREE PROJECT REPORT

Viability of Graph Databases in Pipeline Traceability Systems

David Andersson
Max Villing

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Viability of Graph Databases in Pipeline Traceability Systems

David Andersson
Max Villing

Examiner: Peter Lundin

Department of Computer Science and Engineering
Chalmers University of Technology / University of Gothenburg
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Department of Computer Science and Engineering
Gothenburg 2021

ABSTRACT

Establishing traceability is a continuous problem in modern distributed development environments. In this report the feasibility of using technology agnostic event-based protocols in tandem with graph databases for establishing automated traceability is evaluated. Two different graph databases are evaluated and two different event-based protocols are discussed. It is concluded that it is possible to establish scalable traceability tracking, but that more research is needed for practical applications and that for the traceability to be useful it is not enough to only store the data but it must also be backed by analytics tools.

Keywords: traceability, graph database

Sammanfattning

Hur man fastställer spårbarhet är ett kontinuerligt problem i moderna utvecklingsmiljöer för mjukvarusystem. I denna rapport utvärderas möjligheten att använda tekniska agnostiska händelsebaserade protokoll(spårbarhets protokoll) tillsammans med grafdatabaser för att upprätta automatisk spårbarhet. Två olika grafdatabaser kommer utvärderas och två olika traceability protokoll diskuteras. Det dras slutsatsen att det är möjligt att fastställa skalbar spårbarhet, men att mer forskning krävs för praktiska tillämpningar. Rapporten visar att för att spårbarheten ska vara användbar räcker det inte att bara lagra data utan den måste också stödjas av analysverktyg.

Keywords: traceability, grafdatabas

PREFACE

This report is our thesis report for the bachelor program in computer science at Chalmers University of Technology. The report is written by David Andersson and Max Villing, during the spring term of 2021. We would like to give special thanks to Samuel Alinder and Edvin Agnas at Volvo as well as Jonas Duregård at Chalmers. We would also like to thank the employees at Volvo who helped us during this project.

Table of Contents

1. Introduction **1**

2. Technical Background **4**

2.1 Continuous Integration and Delivery (CI/CD) 4

2.1.1 Continuous Integration (CI) 4

2.1.2 Continuous Delivery (CD) 4

2.2 Software Traceability 4

2.2.1 The Eiffel Protocol 4

2.2.2 The Deep Insight Protocol 5

2.3 Graph Databases 5

2.3.1 Neo4j and Cypher 6

2.3.2 Tigergraph and GSQL 6

2.4 RabbitMQ 6

2.5 Microsoft Azure 6

2.6 Elasticsearch 7

2.7 JSON 7

3. Previous research **8**

4. Method **9**

4.1 Performance Tests 9

4.2 System Construction 12

4.3 Usability Tests 12

5. Results **15**

5.1 Performance Test Results 15

5.2 System Construction 18

5.3 Usability Test Results 21

6. Conclusion **23**

7. Discussion **25**

Bibliography **27**

1. Introduction

As software development continues to evolve, several new practices and methodologies have been created to ease the development process. Continuous Integration (CI) and Continuous Delivery (CD) are modern practices put forth to make software development more flexible and reliable. They entail frequently integrating code changes from different developers (CI) and frequently serving clients with updates (CD). Properly applied, these methodologies allows for a great deal of flexibility for software developers and the ability to quickly identify problems with the codebase [1]. But even if this is handled well a common problem is that there is no traceability in the system, making it hard to track a certain change and what effects it had downstream in the development process. Having full knowledge of all the changes made during development has a wide variety of applications, ranging from knowing what certifications it will need, to being able to estimate the current status of the project, being sure it meets the clients requirements and being able to identify bottlenecks in the development process [2].

While traceability can easily be achieved in more traditional development settings where all work is conducted under one roof and developers work in the same physical location it becomes significantly harder in modern distributed settings where work is conducted over large physical distances. The sheer scale and communication problems inherent to such a setting means the context of a change can easily be “lost” within the development pipeline. A solution to this is extensive manual documentation but this comes at the cost of time and resources [2].

This has become an increasingly pressing issue since the software industry is moving towards more distributed development where every team has a large responsibility for their own development environment, priorities, deliveries and planning. Meaning communication between teams is often difficult as they do not necessarily have a clear understanding of each other's work. Having an automatic system in place could help address this via establishing a clear protocol for what data to communicate and how communication should occur. The system would accomplish this by automatically recording what is occurring in the development pipeline. In order to establish such a tracking process, two things would be necessary. Firstly, a protocol would be needed to establish how tracking data should be recorded. Secondly, the tracking data would need to be stored and easily traversed in a database system.

When databases are brought up in conversation most commonly it refers to relational databases. These types of databases store data in tables, with internal constraints and relationships between tables that ensure data is valid and consistent. Relational databases are traditionally manipulated via the Structured Query Language (SQL). In recent years, however, there has been an increase of non-relational databases that use different means to store data[3]. As these databases do not use SQL they are commonly known as NoSQL databases. One of the foremost of the different types of NoSQL databases is Graph Databases, which store data as graphs, with objects represented as vertices (henceforth “nodes”) and relations between objects as edges (henceforth “relationships”) [4]. Graph databases have a great deal of flexibility when it comes to data analytics and have increasingly been adopted for use in fields such as machine learning [5] and fraud detection [6]. Since graph databases are designed to track how different data points relate to one another they are an attractive option for tracking project events and their causes.

Two protocols that could be suited for generating such data are the Eiffel protocol and Volvo's Deep Insight Protocol. With such a protocol in place it would be possible to track both changes in the development environment and also record what prompted those changes to be made. By translating this data into a Graph Database, it should then be possible to write queries which could show statistics of the entire development history of a project and easily display what prompted any one change. Both the Eiffel and Deep Insight protocols are technology agnostic and should be applicable in any development environment that uses the principles of CI/CD. Furthermore, the protocols can be customized to provide more or less information to fit the needs of any one project.

Given a suitable protocol and database the question then is how to construct a system that establishes system traceability in a way that does not require more effort than existing solutions and is flexible enough to fit different types of projects. An automated system that required more effort than manual systems would be of little use. Care must also be taken to ensure that the implementation of the system is not too complex. If the system is easy to use once implemented but the implementation process is too demanding then the system will do little good. The system therefore will need to be able to handle multiple different types of infrastructure and not be specific to any one setup. Finally, the system must also possess tools that can analyze the stored data and make practical use of it.

The goal of this report was to evaluate if traceability data could be efficiently stored in a graph database. To that end different graph databases were evaluated based on their merits, in particular scalability, before one was selected as the basis for the rest of the testing. Then a system was constructed to facilitate a graph database which could both insert and query traceability data. Finally, some usability tests were conducted to see if traceability data stored in a graph

database could be used efficiently and provide value for the developers. The ultimate aim is that the use of these tools will provide for an automated way to achieve traceability and provide greater understanding regarding the impact of any given change made during the development process in a distributed project.

In order to ensure the scope of this project did not grow beyond the time frame allotted limitations were put in place on the study. Firstly only two different databases were evaluated. Secondly only a limited amount of usability tests were performed and no collection of real-world data occurred, instead existing data already collected by Volvo (with whom this study was conducted) was utilized. The findings here were primarily meant as a proof of concept to prove if full scale software traceability can be stored automatically in a graph database and if this is a viable option at a large scale.

2. Technical Background

In this chapter a number of technologies and tools will be described that were used during the project.

2.1 Continuous Integration and Delivery (CI/CD)

Continuous Integration and Delivery, or CI/CD as they are more commonly known, are two closely related software development methodologies. Their practice ensures that a codebase is frequently built, tested and distributed (often automatically) which gives developers a more efficient workflow as well as the opportunity to find problems earlier in development [1].

2.1.1 Continuous Integration (CI)

Continuous Integration is a methodology where developers integrate their local changes more frequently than in a traditional workflow. By doing so more people are exposed to the code allowing for faster error discovery. This kind of workflow is usually combined with automatic tests to make sure no previous functionality gets lost and the code will behave as it should after new updates [1].

2.1.2 Continuous Delivery (CD)

Continuous Delivery is a methodology where developers strive to cut down the time between software revisions. This allows the codebase to continuously stay tested and up to date with a relatively new release candidate always at hand. This way of developing software reduces many of the risks of more traditional development processes as it's easier to see how well the project is proceeding and if it has potential. This workflow also allows for new features to be brought to the market more quickly [7].

2.2 Software Traceability

Software traceability is a concept where any change in a software project should be stored alongside information relating to what caused the change. By doing so it allows the developers to get an overview of the project which is especially useful in large distributed development environments [8].

2.2.1 The Eiffel Protocol

The Eiffel protocol is an open source project originally developed by the company Ericsson. The protocol consists of multiple JSON schemas on how traceability data should be stored. The goal of the protocol is to achieve full traceability in a CI/CD development environment. This is achieved by using events to represent different stages of software development. When activities or changes occur in a

system using Eiffel they are recorded as events and broadcasted throughout the system, allowing all interested parties to input how they plan to handle the change. Eiffel supports a wide variety of different event types for use in different situations [9].

2.2.2 The Deep Insight Protocol

The Deep Insight protocol is very similar to the Eiffel protocol in functionality but was developed in-house by Volvo. It is still in active development and thus its exact specifications are not fully settled. Similarly to Eiffel, Deep Insight is also based around storing traceability data as events. While Deep Insight is designed to contain traceability data it does so in a looser manner than Eiffel. Compared to Eiffel which formally categorizes many different types of events and mandates how they present its data. Deep Insight uses one general event template instead of multiple different ones.

2.3 Graph Databases

Graph databases are a type of database that stores data in the form of nodes and relations between nodes. For example, a “person” would be a node and if they owned a car, “owns” would be a relation and “car” another node. These nodes and relations can both include additional information, in the person/car example the person could include information such as name and age while the owns relation could store when the relationship was established. Figure 1 below illustrates a simple graph database including a couple of nodes and relations [10].

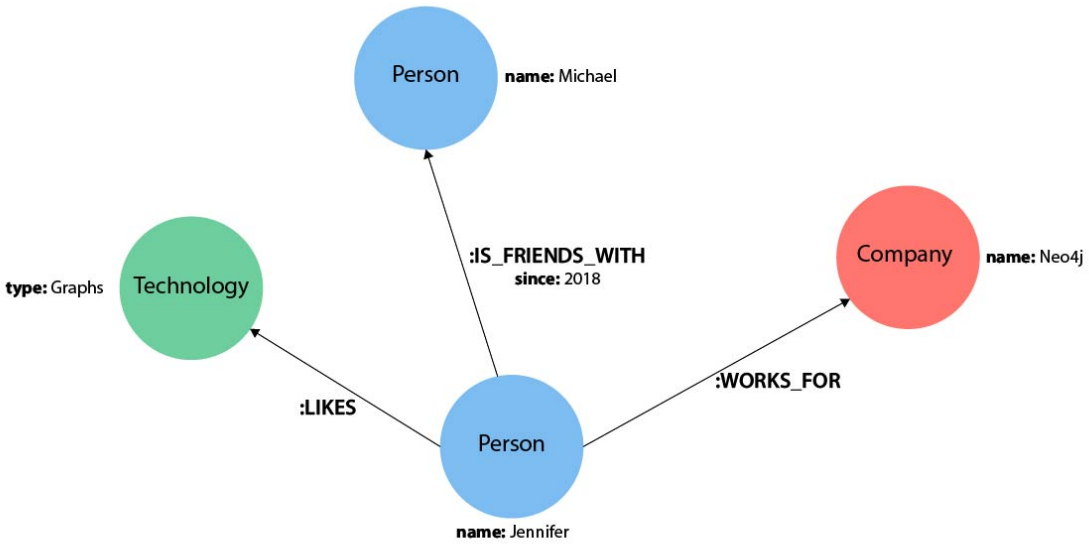


Figure 1: Graph database visualization done in neo4J showcasing a couple of different nodes and relations

Source: [11]

2.3.1 Neo4j and Cypher

Neo4j is one of the most popular graph database systems and is widely used in commercial applications by companies such as IBM and Microsoft. Neo4j uses the query language Cypher to communicate with its databases. Cypher is made with graph databases in mind, which differentiates its syntax from most other solutions based on the Structured Query Language (SQL) [11].

2.3.2 Tigergraph and GSQL

TigerGraph is a graph database management system released in 2017. Unlike Neo4J, TigerGraph is not open source, it is a commercially developed product. TigerGraph was developed with scalability in mind and uses the query language GSQL, developed specifically for TigerGraph. GSQL is built with the intention of being similar to SQL but adapted for graph databases. A distinguishing feature of TigerGraph is its extensive use of parallelism to achieve high performance [12].

2.4 RabbitMQ

RabbitMQ is an open source message broker for use in distributed systems. It serves as a middleware program allowing for message producers (publishers) and message receivers (subscribers) to efficiently communicate with each other without a direct line of communication. The message is instead sent to a queue where those who are subscribed will receive the message. Thus, communication is possible even in cases where neither side is aware of the other's existence [13].

2.5 Microsoft Azure

Microsoft Azure is a common term for many of the cloud services Microsoft offers. This report will however mainly focus on their virtual machine rental service, since it is used to host many of the services and applications used. The way the rental process operates is that the renter selects a virtual machine suited to their needs and how long they plan to use it. The renter then pays a different amount based on the options they picked [14].

2.6 Elasticsearch

Elasticsearch is an open source distributed search and analytics engine. It is useful for sifting through and analyzing large sets of data. Elasticsearch is part of the Elastic Stack, a set of open source tools meant to be used together for data intake, storage, manipulation, analysis and visualization [15].

2.7 JSON

JSON or JavaScript Object Notation is a standardized data format designed to be an easy to use medium for exchanging data between different sources. JSON has a standardized notation based around nesting data in different levels and any data that follows the JSON format can easily be parsed and generated by other programs [16].

3. Previous research

This is not the first work looking at traceability, event-based protocols nor graph databases. Some notable previous works include:

A 2019 study by Rusu and Huang that compared the performance of Neo4J and TigerGraph when it came to executing a variety of queries on differently sized datasets [17]. That study was what led to the evaluation of TigerGraph in this report.

A 2017 study by Ståhl, Hallén and Bosch that looked into the viability of using the Eiffel Protocol for achieving traceability as well as whether traceability was desirable. Their study concluded that not only was there a desire for traceability but that Eiffel could fulfil it [2].

And a 2019 thesis by Hramyka and Winqvist which looked at the process of adopting the Eiffel protocol in a pipeline environment and the development of a proof-of-concept[18]. A modified version of the means with which this study generated data was used when conducting certain tests in this study.

4. Method

The project was organized using a modified version of the development methodology scrum[19]. In this case that meant scrum meetings were held where tasks were handed out and the previous week was discussed. To organize the project Github was used to structure the scrum board and give an overview of what had to be done. As the project only had two project members the organizational models were used quite loosely and more as guidelines. This was done in order to prevent a scenario where more time was spent on administration rather than working on the project.

The project was primarily performed at Volvo but when meeting in person wasn't an option due to covid-19 pandemic, meetings over Discord [20] and Microsoft Teams [21] were held. Both Discord and Teams allowed the meeting participants to share their screens as well as other resources with each other seamlessly.

The software side of this project was primarily written in the programming language Python as it is a language the group is familiar with and also well suited for testing and communicating with the graph databases.

4.1 Performance Tests

The first step of the project was to determine what graph database is most suitable for the application. In this case that meant good query performance when operating on large data sets. After some initial research two databases were up for evaluation, Neo4J and Tigergraph. It was decided that Neo4J would be run through their own desktop application and TigerGraph would be run as a container in a Docker [22] installation. The reason only TigerGraph was running in Docker was because it does not have a Windows version and when Neo4J was attempted to run in Docker several problems occurred.

Two programs were then constructed that could fill the databases with Eiffel events. Eiffel events were used in this case since they could easily be automatically generated in order to get a large amount of data. In order to enter the events into the database the Eiffel JSON schema had to be interpreted into a graph database schema. An example of how an Eiffel event of the type EiffelActivityCanceledEvent is stored in JSON format can be seen in figure 2.

```
{
  "meta": {
    "type": "EiffelActivityCanceledEvent",
    "version": "3.0.0",
    "time": 1234567890,
    "id": "aaaaaaaa-bbbb-5ccc-8ddd-eeeeeeeeeee0"
  },
  "data": {
    "reason": "Made irrelevant by newly scheduled execution."
  },
  "links": [
    {
      "type": "ACTIVITY_EXECUTION",
      "target": "aaaaaaaa-bbbb-5ccc-8ddd-eeeeeeeeeee1"
    },
    {
      "type": "CAUSE",
      "target": "aaaaaaaa-bbbb-5ccc-8ddd-eeeeeeeeeee2"
    }
        ]
}
```


Figure 22: Example of an Eiffel event of type EiffelActivityCanceledEvent stored in JSON format [23].

The interpretation was done by converting the entire event to a node and adding auxiliary JSON objects and arrays as support nodes. These support nodes held all the additional data of the event and the relation to the main node clarified what type of data it was. These event nodes were then linked together with relations matching the corresponding event type. This allowed all event types to share a base schema and made the traversal of the graph easier. Below is an illustration of how the schema looks in TigerGraph, the Neo4J schema is constructed in the exact same way.

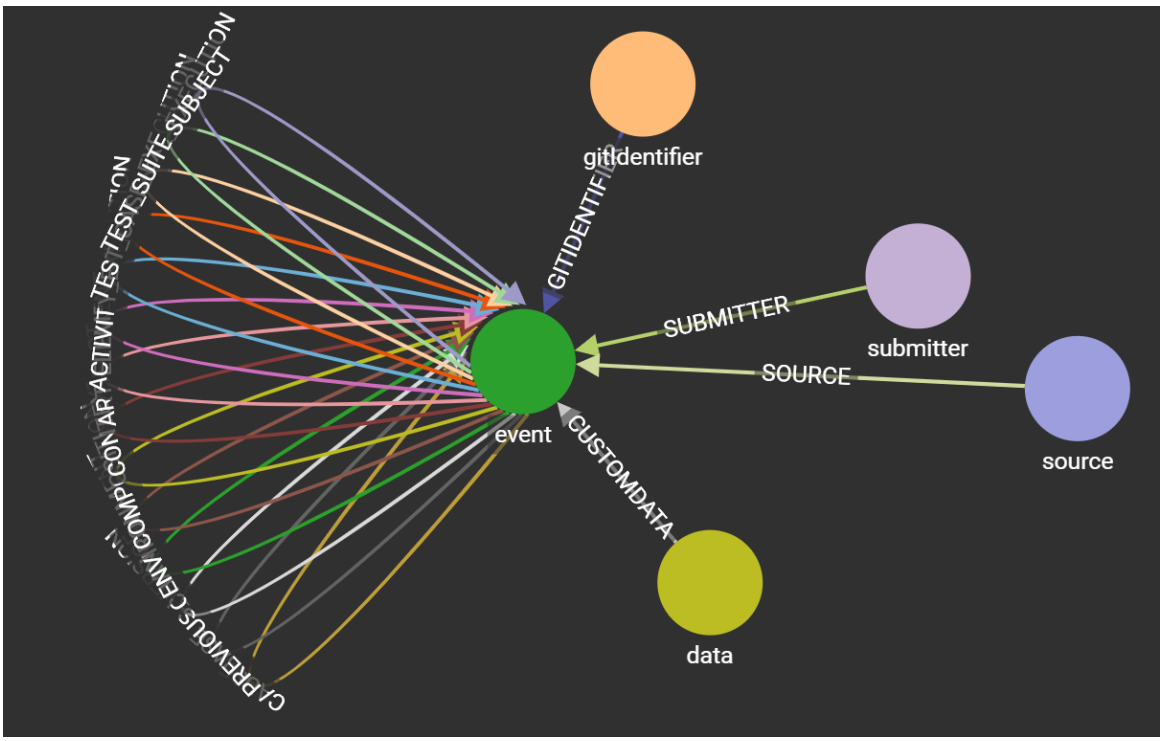


Figure 3: Eiffel Event represented as nodes and edges in TigerGraph

To the left of the event node are all available event types linking back to itself, and to the right are all the available types of data an Eiffel event can be made up of. The programs communicated with the databases through the Python libraries neomodel [24] and pyTigerGraph[25] respectively for Neo4 and TigerGraph. Three documents were then created containing 100 000, 200 000 and 400 000 Eiffel events which were grouped into subgraphs containing roughly 1000 events each. These events were generated by a program based on the event generator provided on the Eiffel-community github [26]. The event generator can both generate and link events together in a realistic way very similar to how it would look like in a real work environment. The insertion programs entered the generated events into their respective databases and timed how long this took. When all the data was in the database a test suite was run multiple times probing different locations in the database. These tests were timed and the results were saved in an excel document for easy comparison.
The test suite includes:

- find a node
- find a subgraph
- find all children of a node
- find all parents of a node
- find all neighbors of a node 1, 2, 5 and 10 steps away

In this case a subgraph refers to all nodes that can be reached by starting at a

given node (including the starting node) and a node's children refers to all nodes who refer to it (and a node is the parent of all its children).
When this testing phase was concluded the project moved on to constructing a traceability system in which the graph database could be further tested.

4.2 System Construction

To start constructing a traceability system it is important to recognize where the data would come from. In this case Volvo already had a system for collecting traceability data formatted in their own traceability protocol, called Deep Insight. So for the purpose of not having to start from scratch their system was used. Deep Insight and Eiffel are basically interchangeable for the purpose of testing the viability of graph databases for a traceability system. They differ in how they represent data and what data they contain but they work on the same principle and if the system proves to work for one it would most certainly work for the other.

Volvo's system mainly consisted of three parts:

- Data collection, where data gets collected from some of their developers.
- Communication, where data was communicated through the message broker RabbitMQ.
- Data Storage, where the data was saved for long term storage and accessed via the use of Elasticsearch.

To get these to integrate with the system two programs were created. The first one was used to insert all existing data acquired through Elasticsearch into the graph database. The second was used to listen for new data from RabbitMQ and insert it as well. Lastly a program was created that could take user input and query the graph database for a result. This program would later be used for the usability tests.

All of these programs were then hosted in the cloud through Microsoft Azure to decentralize the system and allow multiple users to work on the same database.

4.3 Usability Tests

The last step of the project was to perform usability tests. The purpose of these tests were to prove the usability of the system in both what data it can provide and if it could deliver it in a timely manner. It was decided that if the usability tests would take longer than one second to complete they were deemed too slow. Therefore, a test suite was created containing multiple sampling queries which could pick out specific data from the database.

The usability tests were conducted with DeepInsight instead of the Eiffel Protocol. Since Volvo already had a setup that could generate DeepInsight events based on the work of some of their employees, it was decided to go with that protocol for the sake of having real world data. While the mismatch in protocol is not desirable it was ultimately deemed to be acceptable. The performance tests are mainly concerned with testing database performance under high loads and so were mostly protocol agnostic as long as the datasets were sufficiently large and therefore were perfectly suited for our randomly generated Eiffel events. Meanwhile the usability tests are mainly concerned with providing value to the users and are more suited to real data in order to showcase practical results.

The dataset consists of downloaded copies of all DeepInsight events generated by Volvo that were available to us at that time. This encompassed roughly 160,000 events. For the usability tests a test suite was constructed for different usability cases. Compared to the performance tests the execution time was not considered to be significant here. While execution time was recorded, that was only done in order to be sure the queries would not take an inordinate amount of time to perform. Hence a time limit of 1 second was introduced to spot outliers, anything above that would perform significantly worse then what the baseline from the performance test had shown. The use cases that were identified (after consulting with Volvo employees) are listed below accompanied by a sample scenario:

- Finding all nodes without child nodes. This would identify the current status of all projects.
- Find all children of a given node that lack children of their own. Similar to the above, this could be used to identify completed projects and/or dead ends.
- Finding all children of a given node. Can be used to see how a project branches over time.
- Finding all nodes that belong to a given group. Can be used to gain a comprehensive view of a certain subset of the graph.
- Finding which groups have the most nodes in them. Can be used to identify the largest projects in the database.
- Finding all deep insights created by a given author. Can be used to identify who is responsible for which features.
- Finding which publishers most insights were generated on. Allows checking of how many different platforms are used.
- Finding the timestamp difference between two events. Can be used to estimate how long different tasks take.
- Finding all parents of a node. Allows identifying the history behind a given change.
- Find all nodes N hops away from a given node. Allows one to easily gather all nodes with a given amount of relation to the specified node.

- Finding the subgraph that can be reached from a given node. Allows identification of everything that's related to the node, regardless of how strong that relation is.
- Finding how long each event in a series took to complete.

The last use case is the one Volvo was most keen on, since this information would allow them to more easily identify bottlenecks in the pipeline.

The analysis on the practicality of this system could have benefitted from even more tests being conducted. However, considering the given time and resource constraints, it was decided that while the current test suite was not ideal it was acceptable. Had more time and resources been available more use cases would have been devised and more extensive analysis would have occurred.

5. Results

5.1 Performance Test Results

The performance tests provided a lot of insight regarding what database would be most suitable to construct the system around. Figure 3 showcases the difference in execution times between Neo4J and TigerGraph when it comes to executing certain queries. The time differences were found via the use of a Python program that measured the time each database needed to execute the query and compared the two. When performing the tests the latest versions of each respective database were used. At the time this was Neo4J 4.2.1 and TigerGraph 3.1.0. For each query multiple tests were performed and the results were averaged. The tests were performed on 3 different data sets, containing approximately 100,000, 200,000 and 400,000 eiffel events each. The events were randomly generated and were identical on both databases. A complete spreadsheet of all the test results can be found in appendices 1 through 3.

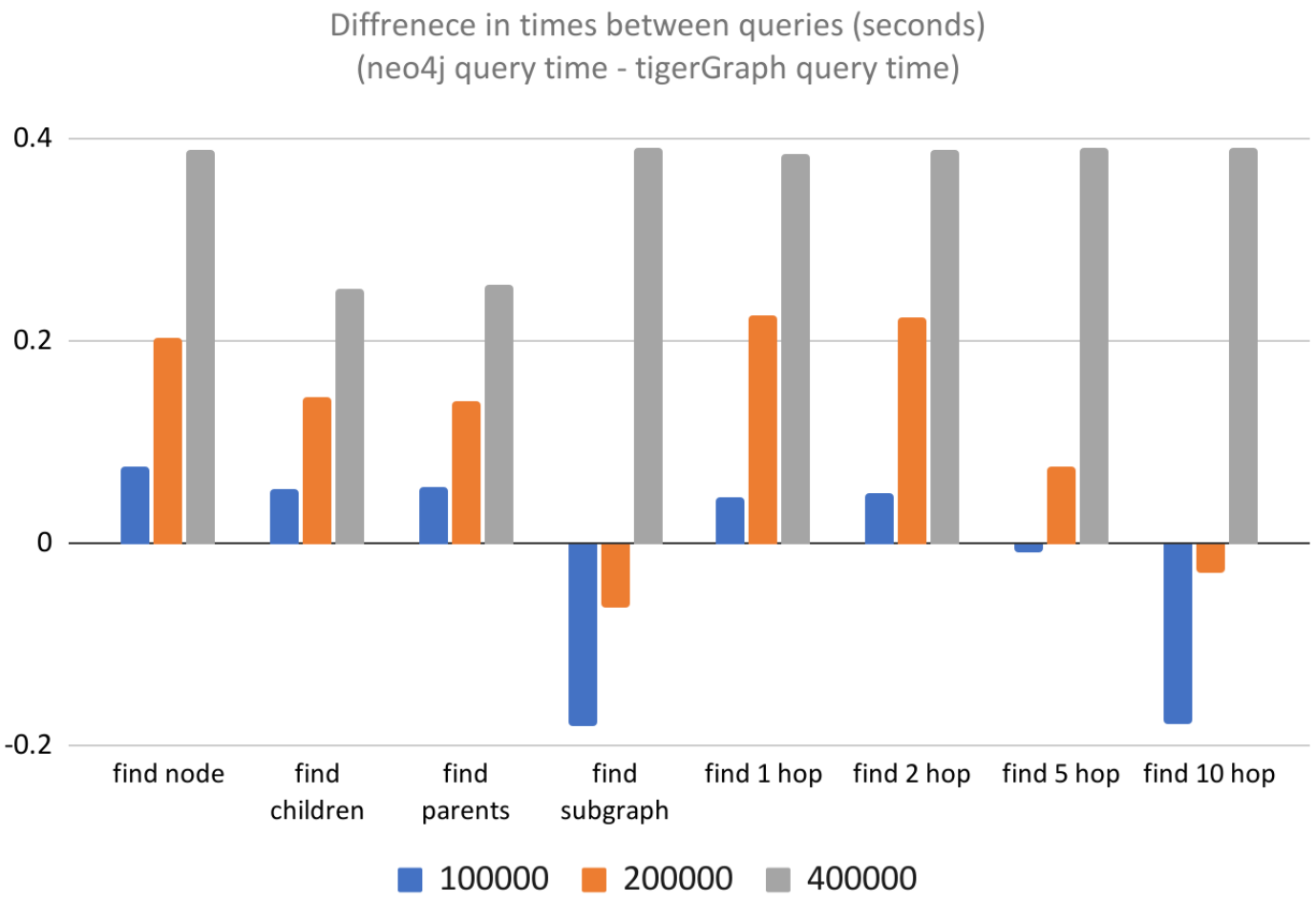


Figure 4: Bar chart showing difference in average query times between neo4J and tigerGraphin in different sized data sets of 100 000, 200 000, and 400 000 Eiffel events. Data can be found in appendix 4, results shown in seconds.

After the databases had been compared to each other the two databases were evaluated in a vacuum to decide if they scaled well when the data set grew larger. If the time to query would grow at a faster rate than linearly with regards to the data set, query times could quickly grow out of control and the system would become unusable. Test results for that can be seen in figure 4 for TigerGraph and figure 5 for Neo4J.

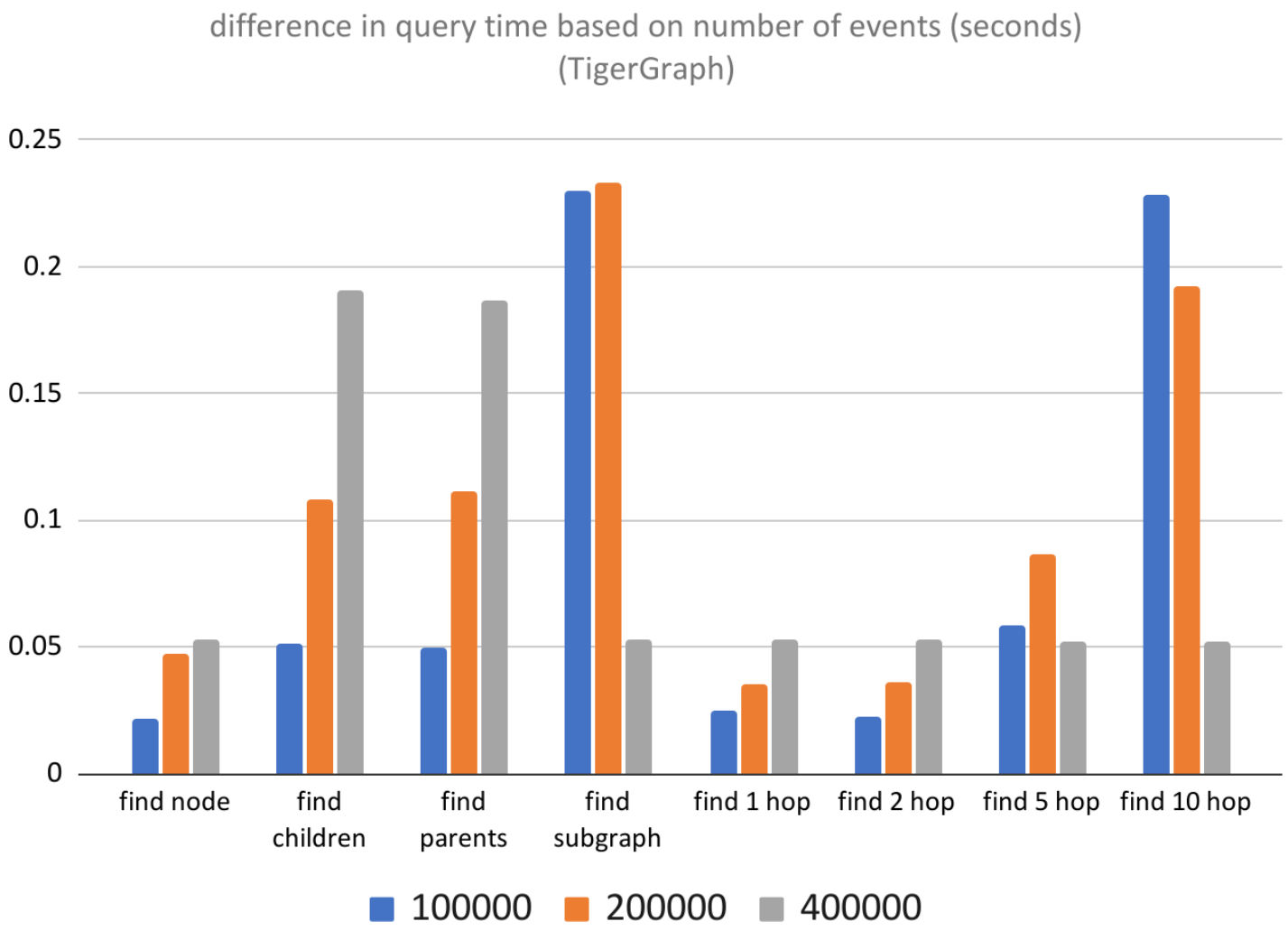


Figure 5: Bar chart showing average query performance for TigerGraph in different sized data sets of 100 000, 200 000, and 400 000 Eiffel events. Data can be found in appendix 4, results shown in seconds.

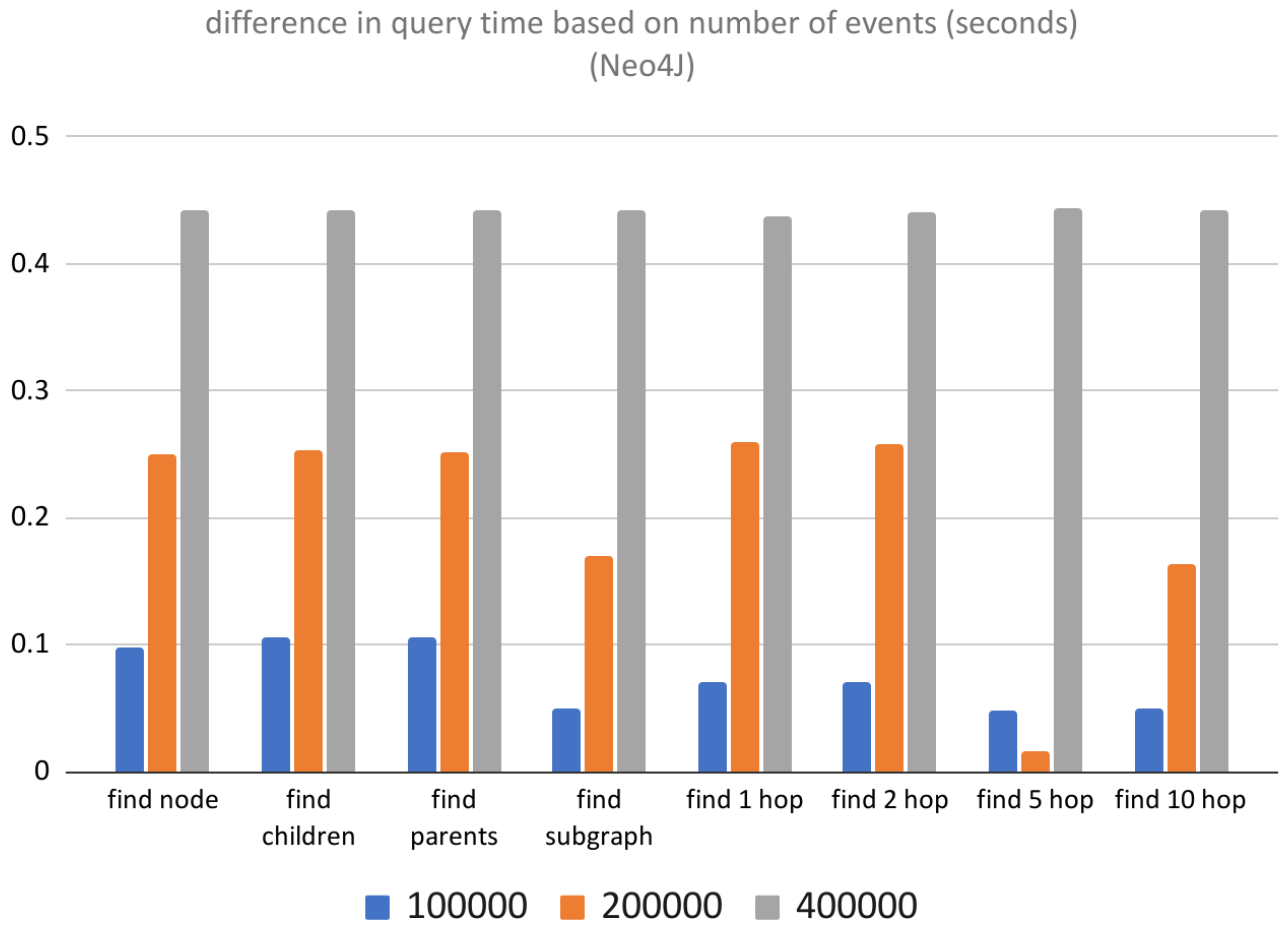


Figure 6: Bar chart showing average query performance for Neo4J in different sized data sets of 100 000, 200 000, and 400 000 Eiffel events. Data can be found in appendix 4, results shown in seconds.

5.2 System Construction

When the performance tests were finished a mockup workflow diagram of a traceability system was created which can be seen in figure 6.

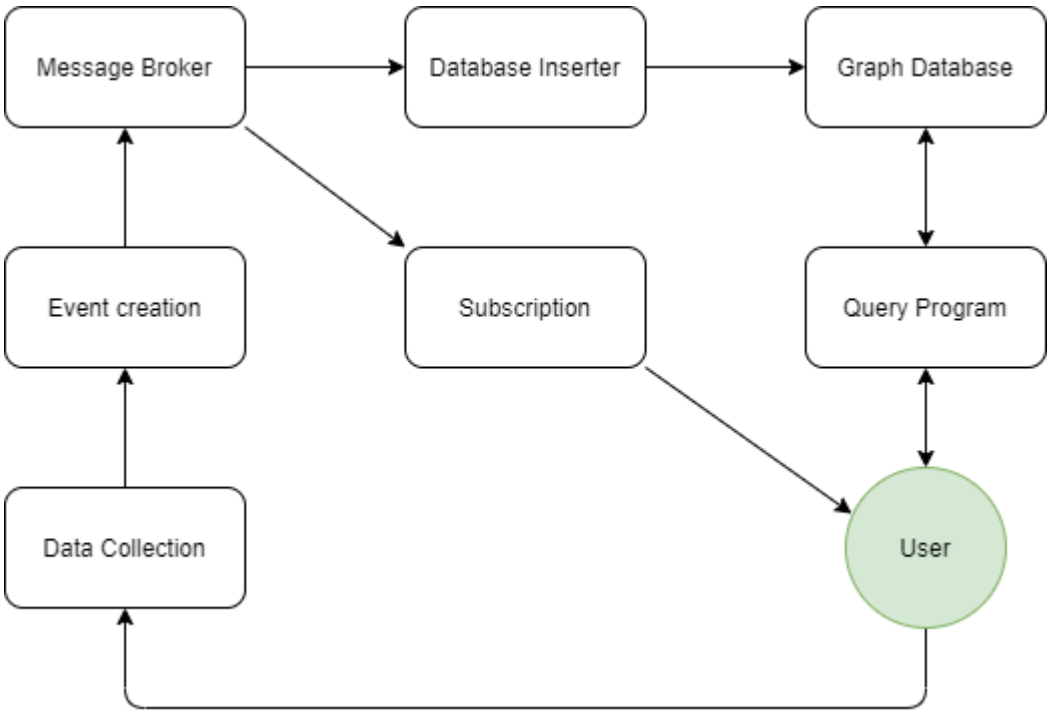


Figure 7: Mockup workflow diagram of a traceability system. The diagram showcases how traceability data will move throughout the system.

This is a general overview of what a complete system could look like. Users working on projects would lead to traceability data being generated in response to their activities. This data would be recorded into events compliant with a protocol such as Eiffel or Deep Insights. The event would then be propagated throughout the system by a message broker, which would ensure the event is both stored in the long term and that any relevant parties are notified. For example, if the event details that a piece of software is ready for testing the message broker would notify the testing team. Users would be able to use a program to query the graph database and receive various kinds of relevant analytics data for the project they are working on. Since the main focus of this project is to see if a graph database could work as a storage medium, an already established data collection system was used and the subscription system omitted. TigerGraph was selected as the graph database for this system due to the performance tests indicating it scaling better than Neo4J. The system created is represented as an infrastructure diagram shown in figure 7.

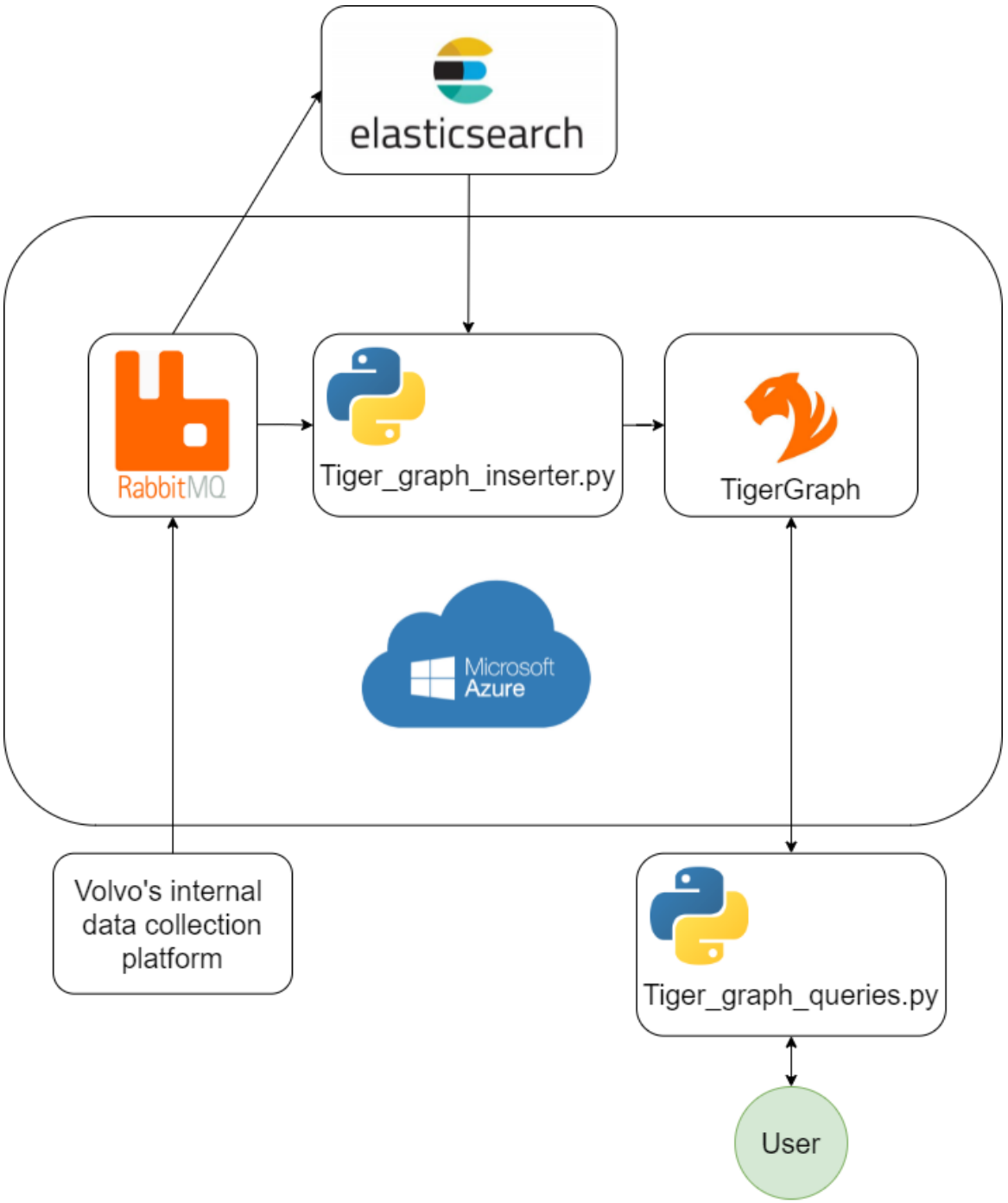


Figure 8: Infrastructure diagram of the traceability testing suite for TigerGraph

Here, traceability data is collected and turned into Deep Insight events by Volvo's own internal data collection platform. The events are then sent to RabbitMQ which forwards them to the rest of the system as well as Volvo's internal data storage solution which could be accessed through Elasticsearch. The system constructed in figure 7 used Elasticsearch to access the already recorded Deep Insight events and used them to fill the graph database with all the previously collected data up until that point. The system then switched to just listening to the rabbitMQ message queue for further data. This was done with a Python script (tiger_graph_inserter.py) which could translate and insert Deep Insight events

formatted in JSON as a usable representation in TigerGraph very similar to how Eiffel events were formatted in chapter 7.1. RabbitMQ, the Python insert script and the Tiger graph database are all hosted on Microsoft Azure virtual machines. This allows the core functionality of the system to easily be accessed from any number of different computers. The final link in the chain was another Python script (Tiger_graph_queries.py) which could retrieve information from TigerGraph on user request. This script used pre-defined queries written in TigerGraphs own scripting language GSQL which had to be pre-installed in the database.

5.3 Usability Test Results

When the traceability system had been designed and constructed it was time to test the usability of the system. This was done by designing and running a number of usability tests whose main purpose was to see if the data stored was of any use to the user. The tests designed were evaluated based on two metrics, if they could provide the data specified and if they could do so in a timely manner. All of the tests listed in chapter 4.3 were constructed successfully and could retrieve the data specified, with all except for one test finishing well under 1 second. All test times can be found in appendix 5 with the average times listed here:

- findInsightsWithoutChildren 1.627s
- findAllChildrenWithoutChildren 0.194s
- findTimeBetweenTwoInsights 0.212s
- findInsightByAuthor 0.428s
- findAllChildren 0.191s
- findAllParents 0.194s
- findMostCommonGroup 0.185s
- findMostCommonSource 0.170s
- findNHops 0.178s
- findSubgraph 0.181s

In particular the test for estimating how much time each event in a series took was proven to work. Volvo had been most keen on gaining that data but had worried that the Deep Insight protocol did not yet record enough data for it to be possible, but testing proved otherwise. While it was impossible to estimate how much time the first event in a series took it was possible to estimate times for all subsequent events by looking and comparing their timestamps.

It should be noted that as these tests were conducted on a TigerGraph installation running in Microsoft Azure, TigerGraph version 3.0.5 was used for these tests. This was the latest version of TigerGraph that could be run on an Azure virtual machine. The major difference between the two versions is that 3.0.5 lacks a feature known as SQL-like syntax which is present in 3.1.0. While

none of the tested use cases needed this feature to run, the query findTimeBetweenTwoInsights could have been made faster if this feature was used.

6. Conclusion

Based on the results from both performance tests it can be seen that the time needed to find a node or set of nodes seems to scale linearly with the size of the database. Other queries however behaved differently between the databases, Neo4J had major performance losses when the data sets grew while TigerGraph was not significantly affected. The tests also showed that when it came to scalability TigerGraph was superior to Neo4J. While Neo4J performed better at low database loads in some cases, TigerGraph had a clear advantage as more and more data was added. If one is looking to establish traceability in a small system, Neo4J may merit consideration. However, based on the metric we were most keen on, namely scalability, TigerGraph was the superior option in terms of performance. Our findings regarding Neo4J and TigerGraph are consistent with previous research that indicate TigerGraph is the more scalable option. A 2019 study [17] found TigerGraph to be vastly superior (by two or more orders of magnitude) than Neo4J when dealing with several different complex queries. While we did not find as significant of a difference as said study, it both used more complex queries and a larger data set from, 1 GB up to 1 TB, in comparison to our tests which was in the range of 0.1 GB to 0.5 GB. We find no reason to suspect that as the complexity and size of the operations in the system grow larger, the advantage of TigerGraph over Neo4J would not continue to widen.

One thing that should be noted though is that Neo4J, owing to its greater popularity and age, enjoys a greater amount of support across platforms. While TigerGraph has an active community, Neo4J has existed for over a decade and has much more public documentation regarding ways to use it. For small teams that lack dedicated resources with which to learn how to use TigerGraph, Neo4J may be a good choice, even if scalability will take a significant hit.

Regarding the usability tests we quickly realized that the primary limitation with making useful queries and analytics tools was not the quantity of data available but what said data described. While many use cases can be conceived, queries to support those use cases can only be supported if the appropriate data has been recorded. The protocol for recording traceability data is hence the main limiting factor when it comes to making useful queries. So picking the right protocol and what it will include is essential to creating a useful system. The more a protocol covers, the more one can do with the data it records. At the same time, it must be kept in mind that the more a protocol requires the less applicable it is across systems, as not all systems will be able to generate the data the protocol requires. The choice of a protocol cannot solely be made on the basis of which one is most helpful for analytics.

Worth noting is that one of the usability tests took way longer than the others and failed the acceptable limit we had set beforehand. The query in question

findInsightsWithoutChildren took almost 2 seconds and almost 4 times longer than the second worst performing query. There could be several possible reasons for this, for example the query created was unoptimized or tigerGraph might not be the best at these kinds of queries. This was longer than the predetermined acceptable time limit, though future optimization may address this..

Based on these findings we can conclude that a graph database can be a suitable option for establishing system traceability. It is however worth mentioning that some complementing functionality had to be constructed outside of the database's native query languages for them to be meaningful to a human reader. An example of this would be that neither GSQL nor Cypher had an intuitive way of comparing timestamps to one another and had to be done in the Python query program. It is also important to keep in mind that the system only works if the graph database can be backed with a suitable data collection and analytic tools.

Graph databases, due to their flexible nature, are a good foundation to build upon, but as is always the case with Big Data, making it useful requires much more than a good storage medium. It should be noted though that our conclusions may not necessarily hold for how a much larger system would behave, a system consisting of millions of nodes in a single subgraph might see worse performance degradation then what our numbers indicate.

7. Discussion

This project is far from a conclusive look at this topic. Handling and analysing large amounts of data is still a new and developing field and distributed development environments are likewise continuously evolving. Due to the sheer breadth of the topic at hand we have had to concern ourselves to a narrow topic and an investigation into a different area may result in sharply different conclusions. Our findings should mainly be taken as a confirmation that graph databases are a viable option for storing traceability data and are worthy of further investigation. Further research would doubtlessly uncover much more about the extent of their applicability and additional ways they could be utilized. In particular, looking into the size of the needed data would be interesting as while we have shown that TigerGraph is more scalable under high load those benefits are not as noticable when it comes to a smaller set of data. If a team knows that they will never have to deal with large amounts of traceability data the advantages of Neo4J may outweigh those of TigerGraph. We also only looked at two traceability protocols, other protocols might have been better or worse suited for use in a graph database. But we believe that due to how flexible graph databases are most if not all traceability protocols should be able to work to some degree.

The methodology used in this project, in particular regarding the performance tests, is not bereft of flaws. Importantly we cannot guarantee that the queries used were optimal when conducting the tests. When databases execute queries they can wildly differ in times depending on how the queries are written. We ourselves noticed this during our testing. At one point a badly written TigerGraph query made it roughly 750x slower than Neo4J. We optimized the queries used as much as possible, but we cannot rule out that there are more optimal queries that would give different results. Additionally the tests were executed via the use of a Python program reliant on external functionality to execute the queries. The use of this software may have introduced some bias favoring one database over the other. Finally, it should be noted that Neo4J was installed and run on the test computer while TigerGraph was used via a Docker container. These different environments may have favored one system over the other. Nevertheless, we believe that, considering the scale of the data sets we used, any bias would not be significant enough to tilt the results. The tests showed a clear and consistent lean in the results and were considered sufficiently objective to make a final decision.

Regarding TigerGraph it should be noted that while the tests were mainly conducted on a Docker version of TigerGraph the system construction runs TigerGraph on Azure. In most regards the difference is immaterial however Docker runs TigerGraph 3.1.0 and Azure uses 3.0.5. TigerGraph 3.1.0 is a later version of the program and so supports additional features, including SQL-like statements which allows for more flexible queries. Some of the tests were written to use these and when converted to run on they lost these features and had to be rewritten. While most functionality was retained the rewritten queries are less efficient.

During the course of this project we considered whether there were any ethical implications or questions regarding this research. We were however unable to find any.

Likewise we also considered the topic of sustainability. And besides general platitudes such as more efficient systems consuming less power we were unable to identify any way in which this research would affect sustainable development.

Bibliography

[1] M. Virmani, "Understanding DevOps & bridging the gap from continuous integration to continuous delivery," Fifth International Conference on the Innovative Computing Technology (INTECH 2015), 2015. [Online]. Available: https://ieeexplore.ieee.org/document/7173368 , [Accessed 05/12-2021]

[2] D. Ståhl, K. Hallen, J. Bosh, “Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework”, Empirical Software Engineering, vol. 22, ss. 967-995, 2017, doi: https://doi.org/10.1007/s10664-016-9457-1 , [Accessed 05/12-2021]

[3] Microsoft, “NoSQL-databas – Vad är NoSQL?”, 2021. [Online]. Available: https://azure.microsoft.com/sv-se/overview/nosql-database/ , [Accessed 05/12-2021]

[4] Neo4J, “What is a Graph Database?”, 2021. [Online]. Available: https://neo4j.com/developer/graph-database/ , [Accessed 05/12-2021]

[5] F. Eaves, “Moving Toward Smarter Data: Graph Databases and Machine Learning”, “Database Zone”, Aug. 2020. [Online]. Available: https://dzone.com/articles/graph-databases-machine-learning , [Accessed 05/12-2021]

[6] G.C.G van Erven, M. Holanda , R.N. Carvalho, “Detecting Evidence of Fraud in the Brazilian Government Using Graph Databases” in “Advances in Intelligent Systems and Computing”, vol 570, Á. Rocha , A. Correia, H. Adeli, L. Reis , S. Costanzo , Springer, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-56538-5_47 , [Accessed 05/12-2021]

[7] P. Swartout, “Continuous Delivery and DevOps : A Quickstart guide”, Packt Publishing, Limited, 2012. [Online]. Available: https://ebookcentral.proquest.com/lib/chalmers/reader.action?docID=1069746&query= , [Accessed 05/12-2021]

[8] J. Huang, O. Gotel, A. Zisman, “Software and Systems Traceability”, Great Britain: Springer, 2012. [Online]. Available: https://link-springer-com.proxy.lib.chalmers.se/book/10.1007%2F978-1-4471-2239-5#about , [Accessed 05/12-2021]

[9] Eiffel Community, “Eiffel”, 2021. [Online]. Available: https://eiffel-community.github.io/ , [Accessed 05/12-2021]

[10] M. Besta, E. Peter, R. Gerstenberger, M. Fischer, M. Podstawski, C. Barthels, T. Hoefler, “Demystifying Graph Databases: Analysis and Taxonomy of Data Organization, System Designs, and Graph Queries”, 2019. [Online], Available: https://arxiv.org/abs/1910.09017 , [Accessed 05/12-2021]

[11] Neo4J, “Getting Started with Cypher”, 2021. [Online]. Available: https://neo4j.com/developer/cypher/intro-cypher/ , [Accessed 05/12-2021]

[12] A. Deutsch, Y. Xu, M. Wu, V. Lee, “TigerGraph: A Native MPP Graph Database”. 2019. [Online], Available: https://arxiv.org/abs/1901.08248 , [Accessed 05/12-2021]

[13] RabbitMQ, “RabbitMQ”, 2021. [Online]. Available: https://www.rabbitmq.com/ , [Accessed 05/12-2021]

[14] Microsoft, “Linux Virtual Machines Pricing”. 2021. [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/ , [Accessed 05/12-2021]

[15] Elasticsearch, “What is Elasticsearch”. 2021. [Online]. Available: https://www.elastic.co/what-is/elasticsearch , [Accessed 05/12-2021]

[16] JSON, “Introducing JSON”. 2021. [Online]. Available: https://www.json.org/json-en.html , [Accessed 05/17-2021]

[17] F. Rusu, Z. Huang, “In-Depth Benchmarking of Graph Database Systems with the Linked Data Benchmark Council (LDBC) Social Network Benchmark (SNB)”. 2019 . [Online]. Available: https://arxiv.org/abs/1907.07405 , [Accessed 05/12-2021]

[18] A. Hramyka, M. Winqvist, “Traceability in continuous integration pipelines using the Eiffel protocol”, Malmö university, Malmö, Sverige, 2019. [Online]. Available: http://muep.mau.se/bitstream/handle/2043/29176/Alena%20Hramyka%20Martin%20Winqvist.pdf?sequence=1&isAllowed=y , [Accessed 05/12-2021]

[19] Scrum, “WHAT IS SCRUM?”, 2021. [Online]. Available: https://www.scrum.org/resources/what-is-scrum/ , [Accessed 05/12-2021]

[20] Discord, “What makes Discord different?”, 2021. [Online]. Available: https://discord.com/why-discord-is-different , [Accessed 05/12-2021]

[21] Microsoft, “What is Microsoft Teams?”, 2021. [Online]. Available: https://support.microsoft.com/en-us/topic/what-is-microsoft-teams-3de4d369-0167-8def-b93b-0eb5286d7a29 , [Accessed 05/12-2021]

[22] Docker, “What is a Container”, 2021. [Online]. Available: https://www.docker.com/resources/what-container , [Accessed 05/12-2021]

[23] Eiffel Community, “EiffelActivityCanceledEvent”, 2021. [Online]. Available: https://github.com/eiffel-community/eiffel , [Accessed 05/26-2021]

[24] R.Edwards , “Neomodel documentation”, 2021. [Online]. Available: https://neomodel.readthedocs.io/en/latest/ , [Accessed 05/12-2021]

[25] pyTigerGraph, “pyTigerGraph”, 2021. [Online]. Available: https://pytigergraph.github.io/pyTigerGraph/ , [Accessed 05/12-2021]

[26] Eiffel Community, “Eiffel”, 2021. [Online]. Available: https://github.com/eiffel-community/eiffel , [Accessed 05/12-2021]

Appendix 1. Performance Test Results: 100,000 Events

find node		find children		find parents		find subgraph	
neo4j	tigerGraph	neo4j	tigerGraph	neo4j	tigerGraph	neo4j	tigerGraph
0.07924509048	0.018338	0.08399438858	0.051656	0.08199691772	0.046525	0.001994609833	0.20263
0.07599663734	0.017256	0.07653403282	0.033445	0.08561038971	0.055459	0.001999139786	0.211
0.07546401024	0.022466	0.07499909401	0.049414	0.08638381958	0.045641	0.0009644031525	0.194767
0.07899999619	0.019886	0.07699751854	0.045342	0.08426046371	0.043173	0.001593351364	0.205048
0.1060059071	0.021677	0.09499287605	0.045753	0.1029992104	0.042718	0.002001047134	0.196053
0.1158883572	0.018723	0.1049575806	0.055592	0.1015241146	0.062129	0.002000570297	0.245473
0.08230876923	0.041898	0.08900594711	0.055611	0.09699440002	0.051455	0.00200009346	0.229896
0.08099842072	0.027001	0.08600354195	0.064029	0.09299468994	0.060863	0.0009932518005	0.282851
0.07898449898	0.018955	0.08200359344	0.047342	0.07803225517	0.041857	0.002025127411	0.213753
0.07605338097	0.019358	0.07594823837	0.048845	0.07701158524	0.048155	0.001988887787	0.218697
0.1189949512	0.021003	0.1071946621	0.045999	0.1049969196	0.045727	0.03200078011	0.212787
0.0929877758	0.025527	0.09499740601	0.046946	0.09100246429	0.040663	0.02699708939	0.207682
0.07599854469	0.031276	0.08900117874	0.04645	0.08199548721	0.042218	0.02800107002	0.203374
0.07599639893	0.028511	0.0769970417	0.044383	0.08501195908	0.044992	0.0207221508	0.196569
0.08299541473	0.017	0.07900190353	0.042331	0.08921217918	0.04446	0.02003693581	0.191017
0.08102464676	0.015646	0.08111071587	0.063059	0.08199954033	0.047134	0.01799893379	0.212835
0.07800340652	0.017425	0.08567023277	0.042696	0.08600020409	0.044383	0.01900172234	0.203705
0.07599949837	0.01319	0.09510207176	0.048602	0.09929442406	0.042743	0.01999282837	0.199484
0.07800388336	0.029533	0.08399915695	0.046056	0.08410525322	0.044406	0.01698756218	0.198255
0.08299803734	0.02603	0.2113423347	0.048537	0.07885789871	0.042986	0.0199944973	0.204422
0.1068394184	0.021564	0.1185958385	0.044062	0.1109972	0.03943	0.05273747444	0.232244
0.1039972305	0.019254	0.08400154114	0.047447	0.08798766136	0.04653	0.04400682449	0.24561
0.07399821281	0.022002	0.09477996826	0.086654	0.08798313141	0.081001	0.0424516201	0.300027
0.08713984489	0.020001	0.06925535202	0.065619	0.07000112534	0.046497	0.02899241447	0.22089
0.08200001717	0.019006	0.07799601555	0.042994	0.07900357246	0.050079	0.03300094604	0.220306
0.07899785042	0.020083	0.07898974419	0.046001	0.08000826836	0.044519	0.02798938751	0.232144
0.09299826622	0.019523	0.08301591873	0.052035	0.07698345184	0.05654	0.0309984684	0.230119
0.08699011803	0.019515	0.07199859619	0.045051	0.07996726036	0.057514	0.0259988308	0.269906
0.08199715614	0.020003	0.07603907585	0.04878	0.07695293427	0.051705	0.02699828148	0.268152
0.0769970417	0.019	0.07999992371	0.059573	0.07699680328	0.053525	0.02795910835	0.231976
0.10155797	0.021001	0.1117472649	0.045005	0.1120023727	0.044582	0.05800056458	0.220382
0.09701013565	0.021245	0.0859670639	0.045003	0.09401035309	0.045509	0.05513739586	0.221609

0.08065724373	0.020003	0.09399747849	0.045518	0.08900356293	0.045518	0.04199624062	0.229354
0.08802103996	0.019999	0.08898639679	0.052185	0.09400343895	0.054036	0.05200076103	0.24616
0.07800555229	0.019	0.07754158974	0.045058	0.07987833023	0.045553	0.03904247284	0.215699
0.09600138664	0.020641	0.06914663315	0.044523	0.07400107384	0.042522	0.03799700737	0.220804
0.08500146866	0.019999	0.09920883179	0.045999	0.09500193596	0.043564	0.0360019207	0.246171
0.07301521301	0.024996	0.08010673523	0.059097	0.08312225342	0.052001	0.0390021801	0.248745
0.07703185081	0.019994	0.07700061798	0.058061	0.08842277527	0.053028	0.03988289833	0.275193
0.1040039063	0.019745	0.1495292187	0.056002	0.1685738564	0.056035	0.04799509048	0.242561
0.1761233807	0.03	0.2673134804	0.063082	0.2510061264	0.055592	0.2540032864	0.266035
0.1716995239	0.027002	0.1850066185	0.080002	0.184497118	0.07183	0.2164883614	0.275114
0.1957836151	0.020994	0.3199088573	0.050524	0.3169808388	0.063081	0.2689511776	0.255336
0.179497242	0.024877	0.1807193756	0.053765	0.1873688698	0.047	0.1210048199	0.230573
0.1595118046	0.019001	0.1395375729	0.056517	0.2045874596	0.065571	0.1355040073	0.297743
0.19486022	0.021518	0.187877655	0.051406	0.2160069942	0.049754	0.1684365273	0.254615
0.1490077972	0.019013	0.1330096722	0.045996	0.1440050602	0.049515	0.1130049229	0.222031
0.08401632309	0.021353	0.0846760273	0.04851	0.08600139618	0.049637	0.05800223351	0.208261
0.07300066948	0.018001	0.08291554451	0.049803	0.07901978493	0.048205	0.06717586517	0.217906
0.08093452454	0.018002	0.07900524139	0.045033	0.07897830009	0.046994	0.06169104576	0.212143
Average		Average		Average		Average	
0.098192873	0.02154068	0.1055545473	0.0510278	0.1064727497	0.04989108	0.04983492375	0.22976214
Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)	
0.076652193		0.05452668731		0.05658166971		-0.1799272163	

find 1 hop		find 2 hop		find 5 hop		find 10 hop	
neo4j	tigerGraph	neo4j	tigerGraph	neo4j	tigerGraph	neo4j	tigerGraph
0.09122347832	0.027173	0.000999212265	0.01009	0.002002954483	0.057048	0.002001523972	0.224273
0.07679510117	0.016703	0.002003908157	0.016353	0.00200009346	0.052428	0.001001119614	0.201618
0.08466506004	0.032109	0.001997470856	0.01951	0.00200843811	0.042375	0.001000404358	0.206847
0.0626616478	0.022208	0.001011610031	0.020903	0.001003742218	0.037179	0.002001285553	0.209787
0.06086301804	0.025415	0.003000974655	0.020436	0.002006292343	0.045619	0.001996517181	0.235391
0.06069159508	0.01824	0.001995563507	0.028131	0.00200009346	0.05338	0.002003669739	0.289608
0.06091213226	0.027974	0.002000331879	0.017653	0.002001285553	0.051785	0.002000331879	0.243537
0.09239864349	0.014436	0.00200009346	0.030538	0.0009982585907	0.046232	0.002003192902	0.216226
0.080037117	0.019646	0.001998901367	0.020432	0.0009989738464	0.044615	0.001000404358	0.230274
0.06151127815	0.023859	0.002004384995	0.021452	0.002003908157	0.047231	0.002000808716	0.201674
0.08884739876	0.0223	0.1095757484	0.019671	0.05399608612	0.050609	0.03699922562	0.227857
0.06908416748	0.019002	0.08800435066	0.017571	0.02500009537	0.055068	0.02599906921	0.186825
0.06900572777	0.019	0.0870013237	0.011907	0.031001091	0.056535	0.03317046165	0.198632
0.05323529243	0.018147	0.08503198624	0.01058	0.02000951767	0.049898	0.02199864388	0.216501
0.07143115997	0.028273	0.09299874306	0.01572	0.02799963951	0.051994	0.02199935913	0.197882
0.06902694702	0.017917	0.09600543976	0.01675	0.02000141144	0.059344	0.01900291443	0.212162
0.06172800064	0.022273	0.0740339756	0.023147	0.01699995995	0.050762	0.01799893379	0.205713
0.0847132206	0.027355	0.07299828529	0.011486	0.01600074768	0.060947	0.01720046997	0.196639
0.06080579758	0.03301	0.0700006485	0.019399	0.01800107956	0.04974	0.01596164703	0.209651
0.08467435837	0.019799	0.08201527596	0.017355	0.01699924469	0.055968	0.02829051018	0.192288
0.09131360054	0.020002	0.05999994278	0.021997	0.06901454926	0.048999	0.04799962044	0.217733
0.0534901619	0.026999	0.02799916267	0.023002	0.04299020767	0.045336	0.04301357269	0.216298
0.06060910225	0.069045	0.04099869728	0.08452	0.04200005531	0.105601	0.03501319885	0.273494
0.06250190735	0.025047	0.03500008583	0.019463	0.04100441933	0.048103	0.03299880028	0.210723
0.06916666031	0.020009	0.03199791908	0.020967	0.02799630165	0.048046	0.03900313377	0.213517
0.06910133362	0.020001	0.02498078346	0.023999	0.03101086617	0.048522	0.02999544144	0.227555
0.07813715935	0.021519	0.02599740028	0.019523	0.02499651909	0.048527	0.02700138092	0.217786
0.0468351841	0.024997	0.02900099754	0.022533	0.02501487732	0.048	0.02600026131	0.203912
0.06251239777	0.024	0.02800178528	0.02	0.02400422096	0.051	0.02900195122	0.234612
0.07048940659	0.021002	0.03000259399	0.023013	0.02503395081	0.049988	0.02699923515	0.203679
0.08896207809	0.021001	0.09999966621	0.021068	0.05400300026	0.096098	0.0668656826	0.246528
0.06250357628	0.022986	0.1119992733	0.021457	0.07799744606	0.097992	0.06799864769	0.219048
0.06910872459	0.020046	0.0939719677	0.020999	0.04599809647	0.092522	0.05201482773	0.294307
0.06891441345	0.022	0.0880177021	0.026	0.05099773407	0.097001	0.04896259308	0.212572
0.06251168251	0.02017	0.09101247787	0.022015	0.04001188278	0.092505	0.04000020027	0.215123
0.07817196846	0.020524	0.07600140572	0.019034	0.04500389099	0.12004	0.03502130508	0.238618
0.06384086609	0.029524	0.07500433922	0.039998	0.04000115395	0.121555	0.03599953651	0.248519

0.08468866348	0.021001	0.07000041008	0.035568	0.03403401375	0.111458	0.03399753571	0.249694
0.06152319908	0.019001	0.07598996162	0.021	0.04296875	0.10652	0.03800201416	0.287598
0.07954621315	0.023532	0.1178987026	0.02152	0.1090033054	0.109138	0.0610024929	0.269423
0.06759190559	0.020005	0.1948258877	0.021546	0.1865477562	0.043005	0.2569534779	0.286098
0.08462047577	0.027859	0.1730041504	0.024654	0.1809709072	0.028994	0.1750068665	0.235275
0.06900167465	0.025996	0.1953129768	0.024004	0.1810061932	0.030959	0.1647822857	0.242975
0.06653952599	0.024	0.2250788212	0.023	0.1791632175	0.029011	0.20135355	0.260037
0.06899404526	0.027	0.1430130005	0.028043	0.1000013351	0.032055	0.1369640827	0.269107
0.08880734444	0.069523	0.1783266068	0.028516	0.1720461845	0.034998	0.1655879021	0.22719
0.07115244865	0.022002	0.1091685295	0.021045	0.1030085087	0.028005	0.1510055065	0.225521
0.06902837753	0.023	0.07600045204	0.020526	0.06347727776	0.026783	0.06796908379	0.210575
0.07062244415	0.021001	0.08234453201	0.021004	0.06249666214	0.027304	0.06100153923	0.222036
0.07253098488	0.024007	0.08221936226	0.022994	0.05800032616	0.028031	0.05743765831	0.215091
Average		Average		Average		Average	
0.07114257336	0.02463276	0.07139695644	0.0226418	0.04885673046	0.05829706	0.05021167755	0.22796058
Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)	
0.04650981336		0.04875511644		-0.009440329541		-0.1777489024	

Appendix 2. Performance Test Results: 200,000 Events

find node		find children		find parents		find subgraph	
neo4j	tigerGraph	neo4j	tigerGraph	neo4j	tigerGraph	neo4j	tigerGraph
0.3113510609	0.050092	0.2849123478	0.146079	0.2686817646	0.158782	0.1002697945	0.278802
0.2612457275	0.425919	0.2352454662	0.110957	0.3020789623	0.109084	0.1007950306	0.257165
0.1848974228	0.031031	0.2001011372	0.094311	0.2509143353	0.111304	0.08046245575	0.260767
0.2160565853	0.032001	0.2005221844	0.104129	0.184350729	0.104173	0.07813715935	0.289876
0.1845836639	0.033509	0.2154705524	0.130335	0.1846041679	0.136148	0.07813620567	0.263132
0.2075212002	0.031006	0.1781368256	0.099559	0.200527668	0.108144	0.08197212219	0.23356
0.188076973	0.034008	0.2016792297	0.107587	0.1960828304	0.118927	0.07070541382	0.2265
0.2192838192	0.038006	0.1967368126	0.107096	0.1971170902	0.092604	0.05648350716	0.23824
0.1831665039	0.037031	0.2007052898	0.132119	0.1984012127	0.13769	0.05488634109	0.231336
0.2101838589	0.035568	0.214148283	0.128728	0.1734468937	0.110261	0.07506966591	0.2321
0.1966519356	0.034001	0.1987636089	0.127605	0.1875295639	0.106091	0.116771698	0.227513
0.1972310543	0.031993	0.2038984299	0.106756	0.1849381924	0.101129	0.1146390438	0.236109
0.2054386139	0.037049	0.2069189548	0.108667	0.168943882	0.105162	0.111385107	0.242095
0.1910238266	0.032563	0.1808171272	0.131395	0.195145607	0.147789	0.1057729721	0.229103
0.1849832535	0.03105	0.2000381947	0.109967	0.2030053139	0.111139	0.09793972969	0.24214
0.1998391151	0.035039	0.1842331886	0.113273	0.1994044781	0.116739	0.1113545895	0.237948
0.2087767124	0.036523	0.1988933086	0.109575	0.185147047	0.102843	0.1002297401	0.240341
0.1902463436	0.030008	0.1746101379	0.100117	0.1932051182	0.14202	0.1198985577	0.234551
0.2001743317	0.032538	0.196408987	0.098136	0.1882636547	0.103127	0.08055090904	0.24077
0.2890350819	0.030911	0.3218996525	0.088113	0.3105902672	0.093625	0.1731693745	0.245745
0.4009363651	0.029492	0.478107214	0.085032	0.3589978218	0.089554	0.2780742645	0.257156
0.3342003822	0.030557	0.3301022053	0.083555	0.324226141	0.08816	0.2089989185	0.234666
0.3697104454	0.029855	0.3193695545	0.088078	0.3085019588	0.090222	0.1908018589	0.226986
0.3577001095	0.025998	0.3375165462	0.086982	0.3083789349	0.086045	0.2520000935	0.215634
0.2509975433	0.035784	0.2424468994	0.094031	0.2574830055	0.104845	0.1729516983	0.224112
0.2400052547	0.035169	0.238399744	0.109712	0.2641203403	0.094418	0.1994187832	0.213701
0.2297744751	0.032229	0.2530081272	0.111994	0.2352278233	0.139131	0.1627111435	0.218016
0.2559990883	0.031778	0.2703864574	0.087006	0.2440001965	0.090552	0.1737999916	0.223204
0.2390003204	0.031909	0.2531962395	0.086027	0.2520005703	0.0934	0.1757566929	0.20653
0.2620298862	0.028974	0.2629842758	0.161034	0.2559542656	0.153008	0.2030005455	0.216922
0.255875349	0.232917	0.2780001163	0.109581	0.2642478943	0.113358	0.1981568336	0.217215
0.2275218964	0.050596	0.2517280579	0.126999	0.2657322884	0.098859	0.1971437931	0.223831
0.2509989738	0.03324	0.2293424606	0.096622	0.2536873817	0.092657	0.2100007534	0.212282
0.232229948	0.030041	0.2621231079	0.08913	0.2563772202	0.094216	0.1879079342	0.217273
0.2411262989	0.042989	0.2368240356	0.144797	0.2581915855	0.121701	0.2043597698	0.224062

0.2459981441	0.030087	0.2701280117	0.095104	0.2799911499	0.105131	0.2022008896	0.212547
0.2541518211	0.035701	0.2517950535	0.09253	0.287217617	0.094025	0.2124361992	0.216239
0.2289748192	0.029999	0.2589931488	0.112556	0.2545213699	0.131078	0.2031767368	0.221355
0.3226413727	0.029042	0.3192870617	0.096033	0.3282723427	0.121819	0.2214980125	0.219638
0.2415697575	0.143269	0.2882204056	0.101537	0.3133215904	0.113647	0.2764794827	0.218004
0.3285207748	0.039371	0.3140077591	0.096162	0.3690357208	0.095593	0.270945549	0.219932
0.2940056324	0.033514	0.3104822636	0.119225	0.3152444363	0.118667	0.2783207893	0.233763
0.3618686199	0.034044	0.3669919968	0.119383	0.3330571651	0.100598	0.3102135658	0.220921
0.2904956341	0.030002	0.2739953995	0.109135	0.2845139503	0.094638	0.2408902645	0.252535
0.2943832874	0.031454	0.2819981575	0.122074	0.2972457409	0.206745	0.2469990253	0.240216
0.2961206436	0.030889	0.2785162926	0.113938	0.3025579453	0.106558	0.266900301	0.243259
0.2464327812	0.033166	0.2379984856	0.111798	0.2579984665	0.1102	0.2201390266	0.240332
0.248552084	0.029051	0.2385573387	0.103112	0.2170307636	0.095129	0.1959998608	0.237881
0.2172236443	0.031059	0.2583198547	0.099029	0.224562645	0.10258	0.1857550144	0.252595
0.2339949608	0.041534	0.2774546146	0.100681	0.2479991913	0.100583	0.2129890919	0.217584
Average:		Average:		Average:		Average:	
0.249656148	0.04767112	0.2532884121	0.10814762	0.251841526	0.11127796	0.169373126	0.23332368
Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)	
0.201985028		0.1451407921		0.140563566		-0.06395055397	

find 1 hop			find 2 hop			find 5 hop			find 10 hop	
neo4j	tigerGraph		neo4j	tigerGraph		neo4j	tigerGraph		neo4j	tigerGraph
0.25745821	0.035019		0.2216482162	0.037		0.09825944901	0.044999		0.1002669334	0.230624
0.2143244743	0.033998		0.1784715652	0.032523		0.06566643715	0.042011		0.08321523666	0.221079
0.2663071156	0.034045		0.1963956356	0.058833		0.08694934845	0.042999		0.07316231728	0.223247
0.1937170029	0.035518		0.2067973614	0.033529		0.07814216614	0.041566		0.06752991676	0.218668
0.1899333	0.034989		0.2003610134	0.036223		0.06903076172	0.040554		0.05928277969	0.212658
0.1939599514	0.037524		0.1797661781	0.034		0.1009972095	0.0433		0.0775282383	0.221182
0.2002122402	0.03552		0.2001035213	0.031907		0.06776881218	0.039554		0.08401346207	0.208309
0.1848397255	0.032525		0.2426550388	0.033521		0.05324435234	0.040523		0.09376072884	0.224068
0.1779081821	0.036524		0.1975016594	0.039577		0.07200932503	0.04179		0.07814073563	0.219658
0.1801013947	0.034004		0.1910357475	0.037005		0.06895875931	0.041522		0.08335351944	0.220267
0.1928896904	0.03502		0.1759080887	0.034172		0.1164181232	0.038527		0.1002719402	0.226552
0.200189352	0.042046		0.1992714405	0.036029		0.1009151936	0.044041		0.08502912521	0.233111
0.2015142441	0.037056		0.2382481098	0.037236		0.1177709103	0.041038		0.1158947945	0.250779
0.2030022144	0.036999		0.1932439804	0.039523		0.0869550705	0.04252		0.1157970428	0.261082
0.2376778126	0.046197		0.2430820465	0.040311		0.08608579636	0.043052		0.1002731323	0.247616
0.1948699951	0.039073		0.1934847832	0.038519		0.1074008942	0.046527		0.1003069878	0.233763
0.2098779678	0.033529		0.1909868717	0.033181		0.1032881737	0.040043		0.09513187408	0.211669
0.1870684624	0.034198		0.1857638359	0.032505		0.0957775116	0.050837		0.1088006496	0.222347
0.1931397915	0.03499		0.190359354	0.033614		0.1005241871	0.039527		0.1027014256	0.216538
0.3283705711	0.033008		0.2450006008	0.03301		0.1530003548	0.037991		0.1660585403	0.218932
0.5084600449	0.034672		0.5440526009	0.034049		0.3069989681	0.105014		0.3419997692	0.231697
0.2710170746	0.032271		0.3019952774	0.033272		0.2250387669	0.102727		0.2079980373	0.216056
0.3265926838	0.033514		0.3850164413	0.036061		0.183031559	0.106567		0.2543292046	0.221062
0.458687067	0.035035		0.4359414577	0.03198		0.2369990349	0.107513		0.1698203087	0.211128
0.2574002743	0.036512		0.2430210114	0.035		0.1537265778	0.112575		0.1470835209	0.220699
0.2685182095	0.035544		0.2907280922	0.034041		0.1837117672	0.105482		0.1721959114	0.208416
0.2210943699	0.033092		0.2532398701	0.032732		0.1482067108	0.106182		0.1762599945	0.21937
0.262488842	0.031841		0.247543335	0.033		0.1714096069	0.105617		0.1691465378	0.215202
0.2824580669	0.031523		0.2319989204	0.033563		0.1793322563	0.101565		0.1591908932	0.22105
0.2640938759	0.038021		0.2521839142	0.033		0.1601967812	0.108558		0.1951491833	0.219778
0.2340481281	0.034566		0.2484343052	0.038992		0.2095179558	0.180996		0.1907320023	0.213154
0.2583031654	0.034874		0.2640714645	0.037723		0.1950008869	0.182086		0.1890823841	0.224087
0.2497951984	0.03281		0.2584476471	0.038659		0.1909997463	0.17432		0.2091023922	0.219619
0.2459983826	0.033523		0.2561886311	0.037854		0.1690139771	0.176523		0.1655087471	0.218035
0.2494211197	0.040064		0.2546391487	0.038421		0.1859970093	0.173077		0.1670000553	0.219405
0.341391325	0.03303		0.319327116	0.037587		0.1669998169	0.170088		0.2318255901	0.208904
0.264165163	0.033588		0.270806551	0.040424		0.1840062141	0.174202		0.1879985332	0.219942

0.2531888485	0.036053	0.2553293705	0.036036	0.1882510185	0.168559	0.1909980774	0.218735
0.3219981194	0.037705	0.3125133514	0.037537	0.2186214924	0.190862	0.2139976025	0.215687
0.4243843555	0.035999	0.3190362453	0.038049	0.2369997501	0.177168	0.2768120766	0.210841
0.3587248325	0.035537	0.3482375145	0.034989	0.3020019531	0.066033	0.253184557	0.222213
0.3552541733	0.031999	0.3209688663	0.033541	0.2831242085	0.071098	0.2820410728	0.222708
0.4094188213	0.033823	0.3953425884	0.034518	0.2931976318	0.065672	0.2776141167	0.224328
0.2892105579	0.032	0.2983481884	0.032548	0.2565498352	0.066447	0.2480049133	0.218553
0.2811894417	0.032985	0.307308197	0.03617	0.2549910545	0.065	0.2287340164	0.207075
0.2434837818	0.033998	0.2611460686	0.034521	0.233212471	0.063657	0.2367677689	0.224369
0.1978600025	0.032005	0.2219996452	0.03277	0.2008113861	0.069571	0.2179992199	0.215793
0.2123916149	0.033648	0.2408745289	0.032794	0.2203342915	0.067138	0.1989986897	0.221077
0.236577034	0.033054	0.2549104691	0.031835	0.2303602695	0.069666	0.1799938679	0.214265
0.2618513107	0.039041	0.2522051334	0.033523	0.1835854053	0.065513	0.1579890251	0.221242
Average:		Average:		Average:		Average:	
0.2603365517	0.03508218	0.25831882	0.03574814	0.1636278248	0.08684794	0.163761549	0.1925471645
Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)	
0.2252543717		0.22257068		0.07677988478		-0.0287856155	

Appendix 3. Performance Test Results: 400,000 Events

find node		find children		find parents		find subgraph	
neo4j	tigerGraph	neo4j	tigerGraph	neo4j	tigerGraph	neo4j	tigerGraph
0.5230715275	0.076785	0.4637334347	0.281016	0.4536166191	0.212945	0.4739663601	0.051535
0.4364416599	0.054083	0.4752640724	0.222367	0.4520924091	0.229745	0.4468355179	0.058557
0.450350523	0.05553	0.4346525669	0.205033	0.4125845432	0.187471	0.4618487358	0.055172
0.436286211	0.056203	0.4226510525	0.195338	0.4422497749	0.180905	0.4401917458	0.046845
0.4230196476	0.046691	0.4318492413	0.185024	0.4152834415	0.182628	0.4399006367	0.051349
0.4197380543	0.051512	0.4379458427	0.199478	0.4319939613	0.181722	0.4166588783	0.051623
0.417170763	0.053875	0.4374530315	0.176412	0.4266624451	0.177195	0.4275443554	0.057624
0.4311273098	0.054694	0.4265418053	0.192215	0.440766573	0.183456	0.4271397591	0.057148
0.421826601	0.054003	0.4317970276	0.178013	0.4207162857	0.18778	0.4394221306	0.045303
0.4238858223	0.050909	0.4307150841	0.197107	0.4308283329	0.176736	0.4174571037	0.046573
0.4693601131	0.053818	0.4851789474	0.186794	0.4705626965	0.178447	0.4896326065	0.051481
0.4405863285	0.049368	0.4422228336	0.194729	0.4421980381	0.186675	0.4280023575	0.049321
0.437349081	0.051405	0.4276094437	0.185544	0.42237854	0.181237	0.4410769939	0.05644
0.4271466732	0.054586	0.4333589077	0.197756	0.4268300533	0.194865	0.4269194603	0.050211
0.4352319241	0.052073	0.4292373657	0.177263	0.4531803131	0.186486	0.4345152378	0.053502
0.4480564594	0.048741	0.4265623093	0.188513	0.4437983036	0.19464	0.4231216908	0.051967
0.4300332069	0.04961	0.4195365906	0.193763	0.414788723	0.193966	0.4321000576	0.056543
0.4509410858	0.054059	0.4274215698	0.200692	0.4209201336	0.197534	0.4387226105	0.049164
0.4772593975	0.053991	0.4866652489	0.202478	0.4375367165	0.193428	0.423248291	0.052987
0.4448764324	0.051639	0.4187412262	0.186118	0.4412994385	0.191653	0.429046154	0.051987
0.4698603153	0.05584	0.4985439777	0.195735	0.4623980522	0.194214	0.4971907139	0.055314
0.4565355778	0.060007	0.434109211	0.179658	0.4259324074	0.179131	0.4857308865	0.052012
0.4232439995	0.045697	0.4313342571	0.183623	0.4263138771	0.175573	0.4320986271	0.056623
0.4250648022	0.052137	0.4477822781	0.187297	0.4694154263	0.179198	0.5492663383	0.05304
0.4198431969	0.051113	0.4321160316	0.187089	0.4317445755	0.178773	0.4286739826	0.055049
0.4318776131	0.054173	0.4386284351	0.176032	0.4166059494	0.177618	0.4335801601	0.055118
0.4258141518	0.048397	0.4379951954	0.18664	0.4103078842	0.180867	0.438107729	0.047986
0.4251987934	0.053132	0.4653503895	0.184108	0.4134142399	0.178653	0.5355832577	0.055968
0.4426307678	0.053391	0.4212884903	0.185491	0.429813385	0.191761	0.4296095371	0.05322
0.4340679646	0.050463	0.4399273396	0.188534	0.4314465523	0.178946	0.407558918	0.049596
0.4943449497	0.06012	0.4763412476	0.184124	0.4405722618	0.184433	0.4752264023	0.050529
0.439132452	0.05604	0.45388937	0.186289	0.4474327564	0.183091	0.4535448551	0.053511
0.4663658142	0.052149	0.5016088486	0.181484	0.4531495571	0.180568	0.413472414	0.047893
0.4388022423	0.059635	0.4298439026	0.173176	0.424710989	0.179707	0.4554300308	0.056713
0.4258606434	0.049405	0.4375500679	0.174937	0.4466934204	0.190503	0.4173223972	0.050913

0.4920909405	0.052514	0.4503321648	0.188295	0.4492294788	0.181954	0.4804570675	0.048328
0.417828083	0.053477	0.4263188839	0.181921	0.4397881031	0.203321	0.4202272892	0.057826
0.4293739796	0.052087	0.4264347553	0.18744	0.5074210167	0.184625	0.4117801189	0.052301
0.4415750504	0.049336	0.4419250488	0.18244	0.4249002934	0.183211	0.4235134125	0.051092
0.4161677361	0.050193	0.4328939915	0.181852	0.4147837162	0.176936	0.4377734661	0.050402
0.4635090828	0.052721	0.4916400909	0.185558	0.4750397205	0.184887	0.4692194462	0.054566
0.4320459366	0.049222	0.466896534	0.175543	0.4772434235	0.17752	0.4433467388	0.053652
0.4376547337	0.059949	0.4190342426	0.196237	0.4246046543	0.188594	0.4426379204	0.0532
0.4525215626	0.049277	0.4460971355	0.192858	0.6841144562	0.177129	0.4345238209	0.057102
0.4411230087	0.052984	0.4238123894	0.211582	0.4328503609	0.197564	0.4311206341	0.055545
0.4413893223	0.050849	0.4250707626	0.191346	0.4245536327	0.189146	0.4357383251	0.051802
0.4450025558	0.053531	0.4202442169	0.197641	0.4414989948	0.180955	0.4284996986	0.060103
0.4318912029	0.051882	0.4421722889	0.189702	0.4234673977	0.191457	0.4211583138	0.053005
0.4466695786	0.055471	0.4390776157	0.199467	0.4329550266	0.187052	0.4164426327	0.051359
0.4329464436	0.04984	0.4112195969	0.179849	0.4247410297	0.191839	0.4391379356	0.049238
Average		Average		Average		Average	
0.4416838264	0.05317214	0.4419723272	0.19083202	0.4421485996	0.1865748	0.4429064751	0.05276676
Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)	
	0.3885116864	0.2511403072		0.2555737996		0.3901397151	

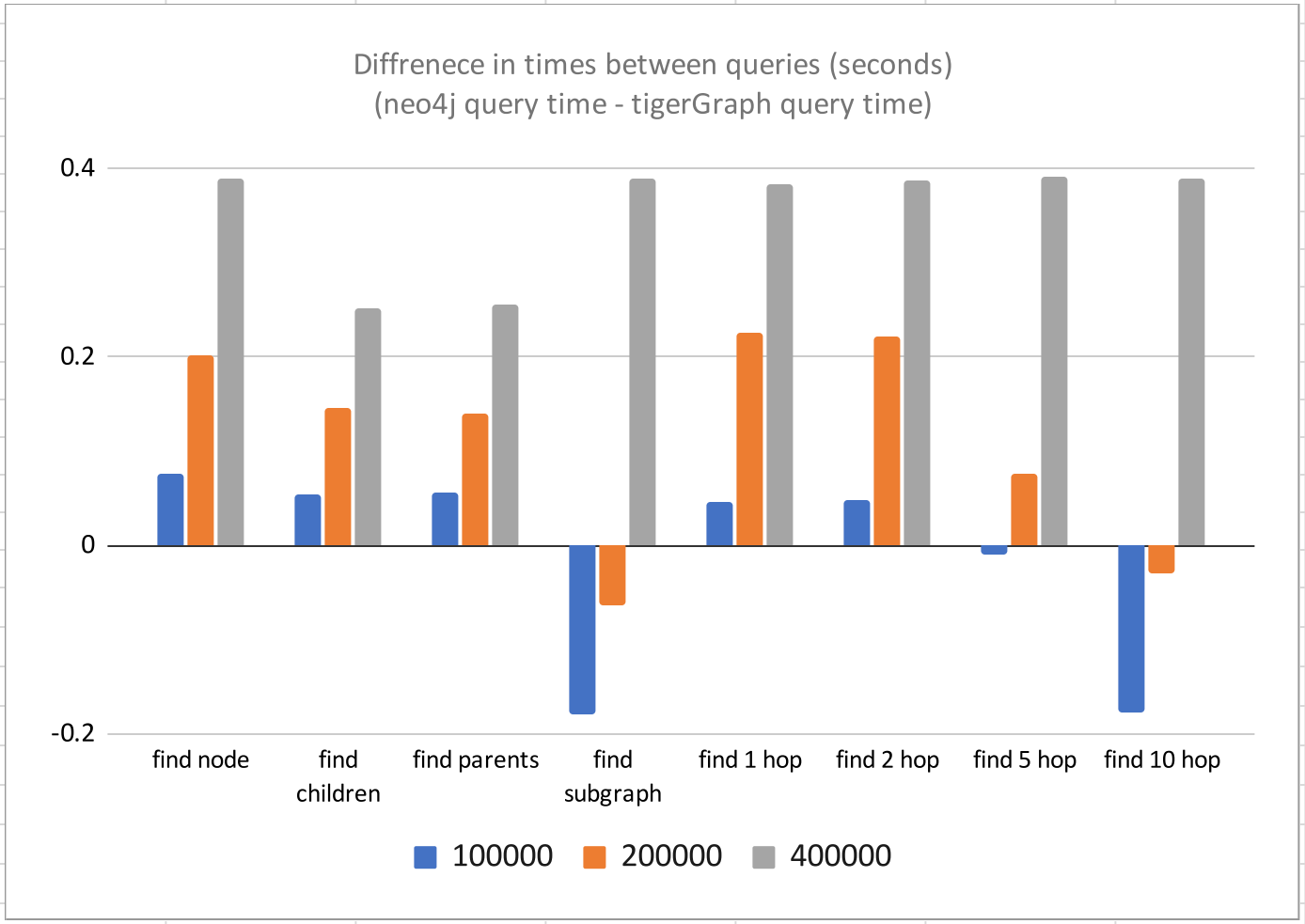
find 1 hop			find 2 hop			find 5 hop			find 10 hop	
neo4j	tigerGraph		neo4j	tigerGraph		neo4j	tigerGraph		neo4j	tigerGraph
0.4874796867	0.065803		0.451038599	0.060947		0.5732722282	0.067589		0.4582295418	0.062815
0.425399065	0.05677		0.4353582859	0.056481		0.4543004036	0.060896		0.4438388348	0.061763
0.4025287628	0.056532		0.4256331921	0.050449		0.4335093498	0.053756		0.4321863651	0.060895
0.4192605019	0.05052		0.4286506176	0.056065		0.4260251522	0.052223		0.4322392941	0.051101
0.4426088333	0.052043		0.4319593906	0.048418		0.4257285595	0.053678		0.4162456989	0.054342
0.4384059906	0.054024		0.4315598011	0.053186		0.4317858219	0.052401		0.4441454411	0.044726
0.4348080158	0.051299		0.4207532406	0.050445		0.4467759132	0.051441		0.4106998444	0.048155
x0.4317779541(	0.050602		0.4368989468	0.052027		0.4456367493	0.046924		0.4192230701	0.053221
0.4253008366	0.060896		0.4159324169	0.050969		0.4348459244	0.053083		0.4426651001	0.048944
0.4315917492	0.054755		0.438816309	0.054017		0.4295148849	0.046496		0.4259631634	0.05564
0.4581980705	0.053928		0.4607470036	0.050406		0.4607329369	0.050421		0.5129585266	0.050389
0.4389531612	0.054204		0.4629569054	0.061393		0.446120739	0.050409		0.44257617	0.047784
0.4179399014	0.049622		0.4430658817	0.049516		0.4289038181	0.052694		0.4378848076	0.05005
0.4318919182	0.056283		0.4420976639	0.049812		0.4285843372	0.05411		0.4673006535	0.051932
0.4223647118	0.053325		0.4317889214	0.051143		0.4277834892	0.051803		0.4323265553	0.051718
0.4279716015	0.047851		0.4337246418	0.056607		0.431987524	0.051242		0.4391078949	0.049505
0.4381849766	0.052482		0.4329469204	0.050002		0.4120724201	0.053843		0.4367640018	0.057999
0.4170026779	0.059864		0.448028326	0.052327		0.4142949581	0.052818		0.5432906151	0.050327
0.4428389072	0.052874		0.4382090569	0.055959		0.47229743	0.054632		0.3966126442	0.054279
0.4358234406	0.049644		0.4323854446	0.054544		0.4475901127	0.052234		0.441838026	0.052642
0.4434843063	0.050556		0.4790101051	0.050127		0.4698774815	0.056084		0.4641771317	0.046649
0.4186673164	0.055288		0.4217252731	0.056815		0.4755706787	0.050672		0.4462981224	0.051701
0.4275536537	0.051103		0.4318141937	0.04917		0.4455704689	0.048702		0.4246373177	0.051505
0.4185788631	0.048604		0.4662020206	0.054024		0.4275381565	0.050921		0.4310648441	0.050937
0.4353165627	0.053843		0.4396266937	0.049048		0.4247801304	0.058338		0.4282677174	0.051664
0.4250366688	0.054389		0.4409821033	0.055029		0.4192993641	0.053065		0.4394638538	0.049348
0.4317705631	0.051029		0.4266147614	0.049888		0.4223911762	0.054666		0.4386854172	0.053309
0.4794447422	0.051869		0.475123167	0.051783		0.4280061722	0.051905		0.422896862	0.050024
0.4379403591	0.047561		0.432104826	0.054024		0.4266211987	0.049971		0.4259669781	0.051113
0.4262127876	0.054648		0.4340724945	0.050842		0.4381482601	0.04993		0.5160758495	0.04878
0.4528326988	0.05008		0.4616949558	0.053203		0.4624710083	0.046132		0.4567992687	0.054559
0.5161888599	0.045701		0.4668092728	0.052876		0.4544730186	0.058692		0.4357156754	0.047208
0.4328110218	0.05422		0.4178218842	0.051778		0.4352834225	0.055747		0.5239481926	0.048733
0.444961071	0.052763		0.4229955673	0.051262		0.4444088936	0.052853		0.4248454571	0.045612
0.4491686821	0.051265		0.4303400517	0.047515		0.4433939457	0.05613		0.4251630306	0.048758
0.4558498859	0.052893		x0.4846158027(	0.053481		0.4654495716	0.052163		0.4698998928	0.04864
0.4325203896	0.052327		0.4458804131	0.059643		0.4358885288	0.046767		0.4263143539	0.054135

0.4332177639	0.053208	0.4160792828	0.047229	0.4318840504	0.048513	0.4288361073	0.058989
0.4250986576	0.053939	0.5940425396	0.050922	0.4513428211	0.057449	0.4538552761	0.054169
0.4401707649	0.057307	0.4210031033	0.052818	0.4336192608	0.047693	0.4238402843	0.051918
0.4652957916	0.049146	0.4679481983	0.060831	0.4624547958	0.045705	0.4610664845	0.057517
0.4141516685	0.049745	0.4498393536	0.053591	0.4373049736	0.047103	0.4353497028	0.057085
0.4250178337	0.049499	0.4509501457	0.05664	0.5010342598	0.052098	0.3995406628	0.054229
0.4331371784	0.053532	0.4303705692	0.05551	0.4351291656	0.052696	0.4297418594	0.054485
0.4311196804	0.056288	0.4142227173	0.049835	0.4337613583	0.052407	0.4272010326	0.051634
0.4261035919	0.048922	0.4314782619	0.058948	0.4305067062	0.053997	0.4441859722	0.051419
0.4415621758	0.048546	0.4250311852	0.054394	0.4610373974	0.047448	0.4397993088	0.057421
0.4333434105	0.050324	0.4120733738	0.053931	0.4339015484	0.051506	0.4381966591	0.055058
0.4315598011	0.048362	0.438192606	0.050622	0.4289062023	0.050981	0.4398386478	0.054999
0.4321291447	0.052233	0.4207792282	0.051771	0.4403135777	0.053473	0.4266326427	0.047724
Average		Average		Average		Average	
0.4367101776	0.05265022	0.440966937	0.05304526	0.4434426069	0.05233	0.4424880171	0.05235098
Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)		Difference(Neo4j - TigerGraph)	
0.3840599576		0.387921677		0.3911126069		0.3901370371	

Appendix 4. Performance Test Diagrams

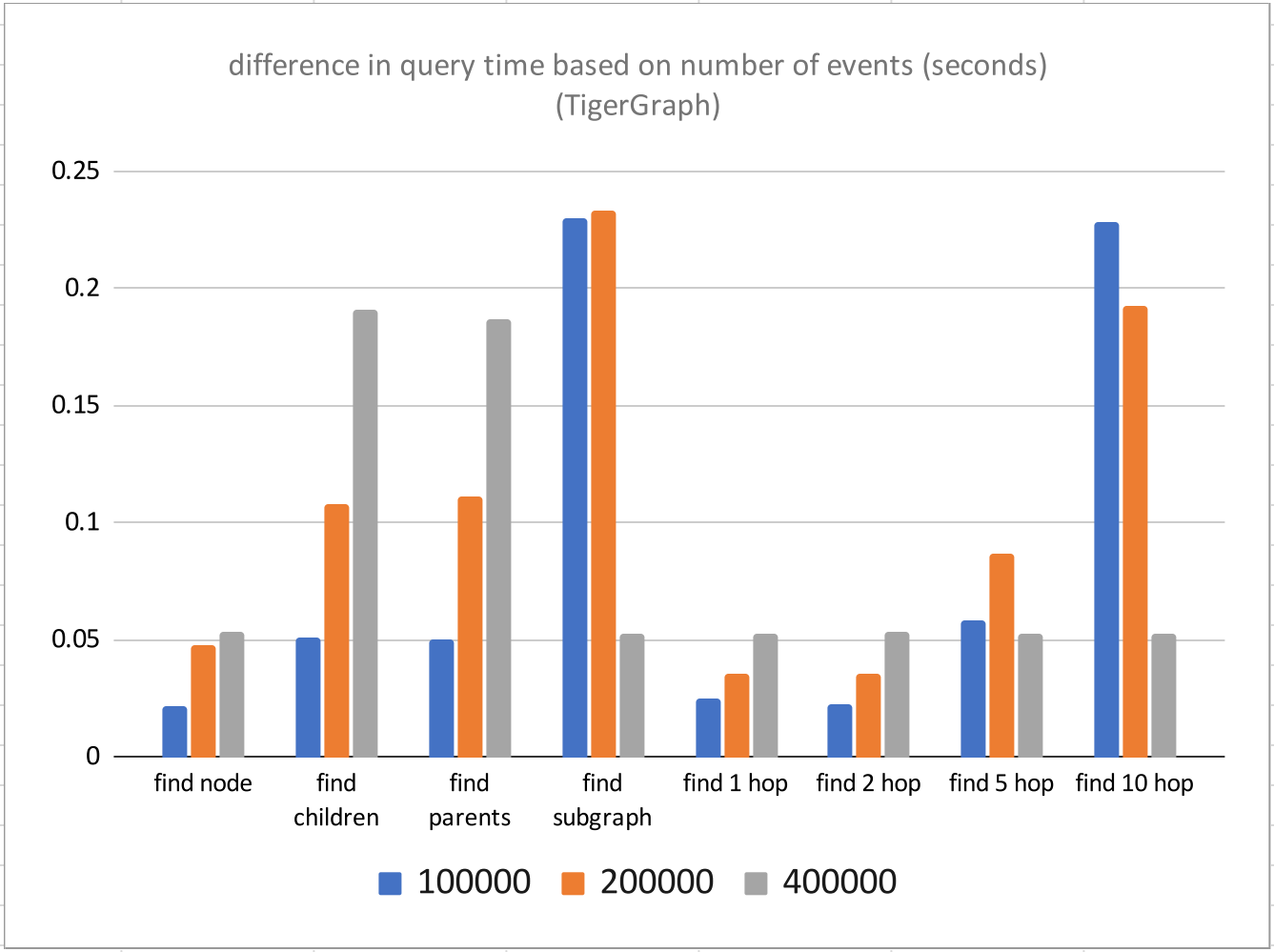
Performance diagrams

	find node	find children	find parents	find subgraph	find 1 hop	find 2 hop	find 5 hop	find 10 hop
100000	0.076652193	0.054526687	0.056581669	-0.179927216	0.046509813	0.048755116	-0.009440321	-0.177748901
200000	0.201985028	0.1451407921	0.140563566	-0.06395055397	0.2252543717	0.22257068	0.07677988478	-0.0287856155
400000	0.388511	0.25114	0.255573	0.390139	0.384059	0.38792	0.391112	0.390137



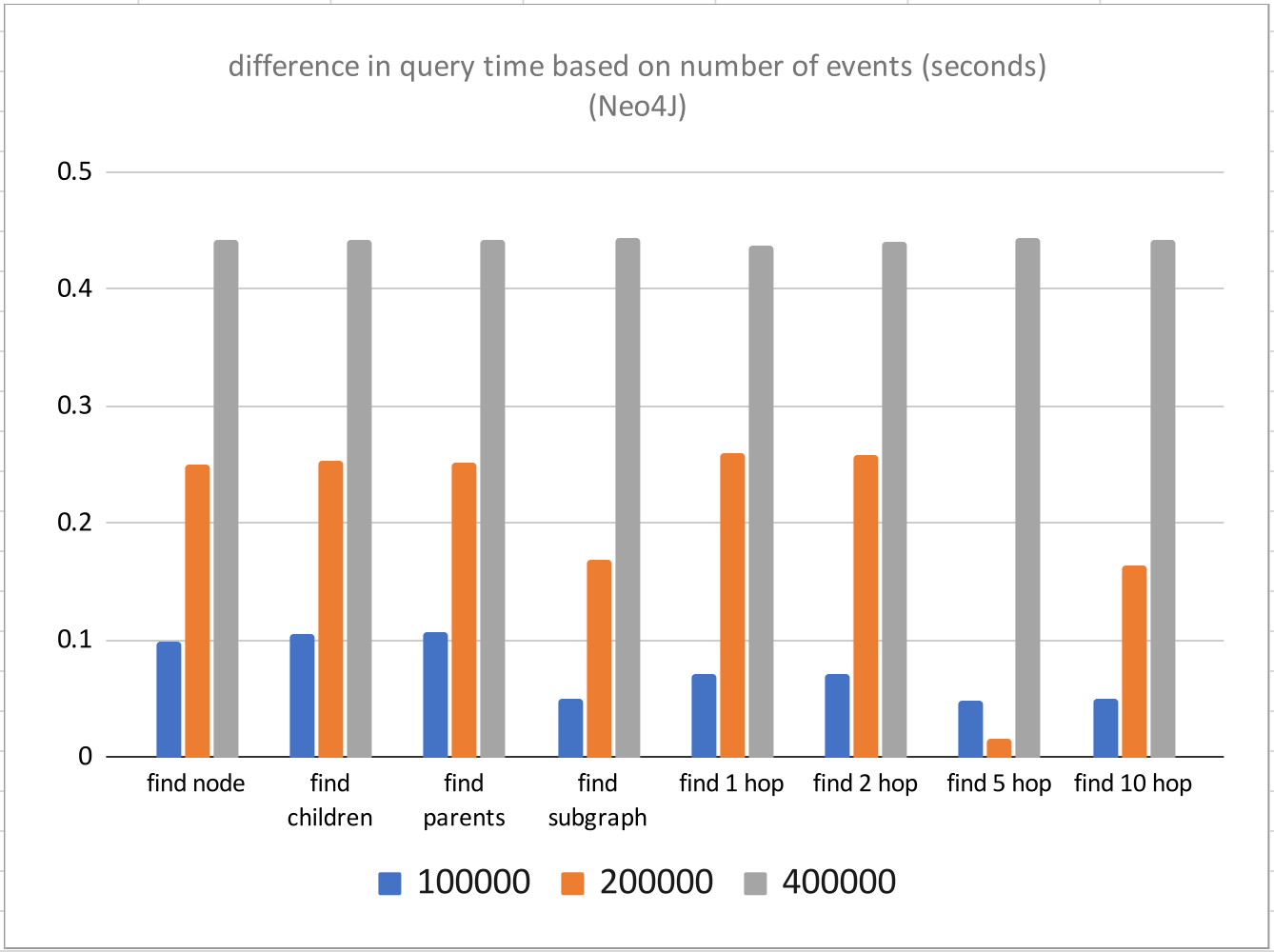
Tigergraph performance

	find node	find children	find parents	find subgraph	find 1 hop	find 2 hop	find 5 hop	find 10 hop
100000	0.02154068	0.05102786	0.04989108	0.22976214	0.02463276	0.02264184	0.05829706	0.22796058
200000	0.04767112	0.10814762	0.11127796	0.23332368	0.03508218	0.03574814	0.08684794	0.1925471645
400000	0.053172	0.190832	0.186574	0.052766	0.05265	0.053045	0.05233	0.05235



Neo4J performance

	find node	find children	find parents	find subgraph	find 1 hop	find 2 hop	find 5 hop	find 10 hop
100000	0.098192873	0.1055545473	0.106472749	0.049834923	0.071142573	0.071396956	0.0485673	0.050211677
200000	0.24965561	0.2532884121	0.251841526	0.169373126	0.2603365517	0.25831882	0.01636278248	0.163761549
400000	0.4416838264	0.4419723272	0.4421485996	0.4429064751	0.4367101776	0.440966937	0.4434426069	0.4424880171



Appendix 5. Usability Test Results: Time Taken

findInsightsWithoutChildren	findAllChildrenWithoutChild	findTimeBetweenTwoInsigh	findInsightByAuthor	findAllChildren
1.5045135021209717	0.22494006156921387	0.1951279640197754	0.4388847351074219	0.19041132926940918
1.4716386795043945	0.193037748336792	0.19563889503479004	0.41860389709472656	0.1913299560546875
1.4827876091003418	0.18024420738220215	0.1888573169708252	0.41016340255737305	0.1881999969482422
1.5787813663482666	0.20462751388549805	0.18233847618103027	0.41429638862609863	0.195281982421875
1.5827631950378418	0.19044756889343262	0.17781305313110352	0.4037296772003174	0.18555855751037598
1.532705307006836	0.195601224899292	0.33997488021850586	0.42287588119506836	0.1995983123779297
1.5165462493896484	0.1822504997253418	0.25237464904785156	0.524219274520874	0.18846487998962402
1.9454221725463867	0.20300889015197754	0.18845725059509277	0.4067118167877197	0.1890418529510498
1.5881109237670898	0.18677783012390137	0.19889497756958008	0.40612292289733887	0.17766833305358887
2.065751791000366	0.18212318420410156	0.2006697654724121	0.4311563968658447	0.20517444610595703
Average:				
1.62690208	0.1943058729	0.2120147228	0.4276764393	0.1910729647

findAllParents	findMostCommonGroup	findMostCommonSource	findNHops	findSubgraph
0.19040298461914062	0.18945813179016113	0.16379594802856445	0.1741483211517334	0.18532896041870117
0.20135164260864258	0.17561578750610352	0.1641712188720703	0.17518019676208496	0.17975401878356934
0.18918395042419434	0.18521904945373535	0.1585235595703125	0.17749834060668945	0.19055438041687012
0.1902143955230713	0.18111109733581543	0.16579151153564453	0.18814349174499512	0.17094945907592773
0.18693232536315918	0.1691875457763672	0.15846562385559082	0.17094779014587402	0.17662835121154785
0.220383882522583	0.20437169075012207	0.18975067138671875	0.19214558601379395	0.1799015998840332
0.1889796257019043	0.1863994598388672	0.1694197654724121	0.1667768955230713	0.2091984748840332
0.18563294410705566	0.18228721618652344	0.19346213340759277	0.17601513862609863	0.1732959747314453
0.20124387741088867	0.18524789810180664	0.16893577575683594	0.17854928970336914	0.16748309135437012
0.18488502502441406	0.19373631477355957	0.16823029518127441	0.1788499355316162	0.18116354942321777
0.1939210653	0.1852634192	0.1700546503	0.1778254986	0.181425786