Karlsruhe Institute of Technology

Benchmarking of Graph Databases - Suitability for the Industrial Environment

Bachelor's Thesis
by

Christian Navolskyi

Chair of Pervasive Computing Systems/TECO
Institute of Telematics
Department of Informatics

First Reviewer:	Prof. Dr. Michael Beigl
Second Reviewer:	M.Sc. Andrei Miclaus
Supervisor:	

Project Period: 01/01/2018 – 30.04.2018

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, den TODO: date

Zusammenfassung

TODO: Zusammenfassung (Deutsch)

Abstract

TODO: Zusammenfassung (Englisch)

Contents

1. Introduction

TODO: Fix url in bibliography

1.1 Problem Statement

TODO: Highlight that current graph database benchmark papers are not covering our field. With the growing digitalisation of the industry more data is available and can be used to improve production processes. The amount of data created depends on the individual use case, but still it needs to be stored to be useful. Since there are multiple databases available it can be difficult to choose the right one for an individual scenario.

1.1.1 Use Case - Industry 4.0

There are multiple analytic algorithms to run on data to extract certain features. In the industry those algorithms play an important role too, but in this thesis we are looking at different aspects of the industrial use case, mainly inserting data and reading data. Though we have no direct partner in the industry for this thesis we try to TODO: No industry but trying our best to get good data / structures / workloads

1.1.1.1 Inserting Data

To digitalise the production processes the data produced by every machine in the production line should be stored for future analysis. And to store that data it needs to be written into a database. Since most factories running 24 hours a day the machines are producing a lot of data during the day. That will be the base load for the underlying database, to store all that data from the production machines.

1.1.1.2 Reading Data

Besides using the stored data for analysis algorithms, simply reading data from the database is another common use case. An example would be to get the time at which a specific product was processes by a specific machine to check if all parameters were set correctly.

1.2 Question

This thesis should give an answer to the question, if graph databases are suitable for an industrial application. Suitable in this case means that the database can withstand the amount of data written to it during production. In chapter 3 section 3.1 we analyse how much data could be written to a database and of which structure that data is. We motivate graphs and the use of graph databases in section 2.1 and section 2.3 respectively and will also concentrate on those in this thesis, because of the structure of data from production.

1.3 Methodology

We will chose the databases to use for our testing from other studies covering benchmarking graph databases to be able to compare the results and look to similarities in behaviour. To evaluate different databases we first will look up existing benchmarks and choose the best fitting one for our research. In the benchmarking program we need to look at the creation of data and how it can be stored and retrieved. The same exact dataset should be used for all databases equally to eliminate the variation that comes with generating data during each benchmark run. Next the workloads should be customisable and be able cover our goals. Also the measurements taken during the benchmark need to be useful for us. The databases should be able to connect to the benchmark in the same way.

1.4 Goal of this Thesis

With this thesis we want to examine whether and if so, how well graph databases are able to stand the load of an production line. Because every manufacturer is different and we cannot cover all scenarios we try to cover the most important parameters so that the suitability for the individual case can be estimated.

1.5 Structure

In chapter 2 we are motivating graph and the use of graph databases. The different kinds of graph databases are explained and an example database which we are testing is mentioned and shortly described. Also in this chapter we are comparing the different available benchmarking programs and their features. NOTE: Maybe mention related work

In chapter 3 the industrial data is modelled and its structure is analysed as well as a reasonable amount of data is determined. Then we are figuring out how a workload could look like in an industrial environment. At last we further analyse out chosen benchmarking program and give an overview of its procedure.

Chapter 4 is focused on the design of the different extensions for the benchmark and also the concrete data structure. For the extension we cover the design of the specific workloads, the design of classes to create and recreate the dataset, the graph workload class managing the graph databases and the graph data and finally the database bindings which are responsible for connecting the database to the benchmarking program.

In chapter 5 the implementation of the single components is described. First we cover the graph data generator which includes the class for creating the graph data as well as the class for recreating it from files. Next the bindings are implemented and their individual adaptions to the benchmark are highlighted. And lastly we explain the graph workload class which is the mediator between the created graph data and the database bindings.

Chapter 6 focuses on running the benchmark and evaluating the results. First, we define our objective during evaluation. Then the configuration of our system is stated, as well as the hardware as the software side. Next the procedure of running the benchmarks sequentially is explained following be the different aspects we are testing. These are grouped into ”maximum load” in section 6.4, ”throughput” in section 6.5 and ”responsiveness” in section 6.6. Each group includes multiple benchmarks in which we changed on variable at a time. The results are presented directly after each benchmark followed by a discussion to interpret the results.

In chapter 7 we draw a conclusion over out work and give the answer to our question from above. Also ideas for future research and development in this field are presented.

Finally in chapter 8 with give a short summary of our work.

2. Background & Related Work

2.1 Industrial Data

Under the term ”industrial data” we understand data that is produced by machines during the production. That could be the current settings of the machine, temperatures or tolerances measured during processing or what product is currently worked on. In chapter 3.1 the possible structure of this data is analysed. NOTE: Clear, that we have to come up with that data?

2.2 Graphs

A graph as the literature tells us [17, p. 89] is a tuple of sets $G = (V, E)$ with $E \subseteq V \times V$. Elements of V are called vertices and elements of E are called edges. The set of vertices has to be not empty, but the edge set can be. In this thesis we are focusing on directed graphs only, although some graph databases are capable of handling undirected graphs too not all are. Also there would be no benefit in using undirected edges since our model also uses directed edges. In general graphs can have labels or weights on their edges as stated in [17, p. 99]. For our purposes we will use labels on the vertices and edges to ease the understanding of our data structure. In section 2.3 we will give reasons why having labels on the graph components is useful.

2.3 Graph Databases

There is a variety of database types available the main categories are SQL and NoSQL databases. A short description of SQL databases would be “A relational database organizes data in tables (or relations). A table is made up of rows and columns. A row is also called a record (or tuple). A column is also called a field (or attribute). A database table is similar to a spreadsheet.” ([6])

NoSQL databases on the other hand are able to store any kind of data in any record, they don’t rely on a specified schema. Also they are able to scale horizontally for the cost of consistency. [19]

Graph databases are a type of NoSQL databases. They use graph theory to store their data as described in 2.2 with vertices and edges. Every vertex has a unique identifier (id) in the database, the edges coming from or going to that vertex and it can have properties assigned to it as key/value pairs. Edges also have a unique id, a start and end vertex and properties just as vertices. [13]

The labels mentioned at the end of section 2.2 can be seen as the properties assigned to a vertex or edge. To map a production line in which the elements like machines and products might have their own real world ids these properties can be used to store these real world ids as a key/value pair. Later a particular machine for example can be looked up by its id. That is critical to find the data stored in the database.

In the following subsections 2.3.1 through 2.3.3 we will discuss the different types of graph databases and give examples of real databases which operate by that type. All databases used in this thesis support the ACID[1] principle with transactions to ensure data consistency.

2.3.1 RDF/Triplestores

First in our list are RDF stores also known as triple stores.

RDF (Resource Description Framework) is a model for data interchange on the Web. It is able to merge data even with different schemas, it also support the evolution of a schema over time. The linking strucutre of the Web is extended by RDF by it using URIs[2] to name relationships and resources connected by those. [11, p. 4]

> This linking structure forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes. This graph view is the easiest possible mental model for RDF and is often used in easy-to-understand visual explanations. ([16])

Triplestores store semantic facts as subject - predicate - object triples, also referred to as statements using RDF. These statements form a network of data, which can also be seen as a graph. [11, p. 4]

2.3.1.1 Apache Jena TDB

“Apache Jena (or Jena in short) is a free and open source Java framework for building semantic web and Linked Data applications. The framework is composed of different APIs interacting together to process RDF data.” ([3])

The TDB component in Jena is responsible to store and query RDF data. [1]

In section 2.4 we will discuss recent studies investigating Apache Jena, among others.

2.3.2 Document Stores

As the name suggests the data model of document stores consist of documents which can have fields without depending on a defined schema[12]. Its aggregates data in those documents and transforms them internally into an searchable form[15].

[1]TODO: Enter Text
[2]TODO: enter text

2.3.2.1 OrientDB

OrientDB is a mix of a document store and a graph store, as stated in their manual "OrientDB is a document-graph database, meaning it has full native graph capabilities coupled with features normally only found in document databases." ([12]) Its designed as a robust, highly scalable database with a wide possible set of use cases. [12]

2.3.3 Graph Stores

Graph stores organise their data as graphs. References with foreign keys known from relational databases are mapped as relationships in graph databases. Each node in the database model contains a list of relationship-records to represent their connection to other nodes. [9]

2.3.3.1 Neo4j

Neo4j is a native graph database and was build as such from the ground up. In their introduction they say "The architecture is designed for optimizing fast management, storage, and traversal of nodes and relationships. In Neo4j, relationships are first class citizens that represent pre-materialized connections between entities." ([10])

2.3.3.2 Sparksee

The user manual describes Sparksee as follows, "Sparksee is an embedded graph database management system tightly integrated with the application at code level." ([14]) Sparksee is implemented in C++ but provides a low level Java API.

2.4 Graph Database Benchmarks

As the need to compare similar programs exists benchmarks are needed to hand results over certain parameters to aid decision making. In the field of graph databases that is no different. There exist multiple benchmarks for graph databases and some are outlined shortly in the following subsections 2.4.1 to 2.4.3. In in section 3.3 we choose a benchmark for our work.

2.4.1 LDBC: Graphalytics

Benchmark specifications, practices and results for graph data management systems are established by an industry council called The Linked Data Benchmark Council. The Graphalytics benchmark uses a choke-point design to evaluate the crucial technological challenges present in system design, one example would be the "large graph memory footprint" as mentioned in [4, p. 2].

Graphalytics uses Datagen to create social network graphs, which are easy to understand for their users [4, p. 3].

The workloads implemented in Graphalytics represent common graph algorithms such as breadth-first search, weakly connected components or single-source shortest paths to name just a few [7, p. 7].

Neo4j was used among others in the study of Capotă et al. [4], which we will refer to in our evaluation in chapter 6.

2.4.2 XGDBench

Is a graph database benchmark for cloud computing systems. It is designed to work in the cloud and in future exascale clouds. XGDBench is an extension of the Yahoo! Cloud Serving Benchmark for graph databases. This benchmark is written in X10, a “programming language that is aimed for providing a robust programming model that can withstand the architectural challenges posed by multi-core systems, hardware accelerators, clusters, and supercomputers” ([5]).

XGDBench also focuses on social networks for their data structure, that is generated by a procedure called Multiplicative Attribute Graph (MAG), see [8] for more information.

It specifically targets read, update and graph traversal operations for its performance aspects [5, p. 366].

This study featured following graph databases which we are also testing Fuseki, Neo4j and OrientDB [5, p. 364]. Fuseki is a SPARQL[3] server providing a HTTP interface to Jena [2], so that research covers three of our four databases.

2.4.3 YCSB

The Yahoo! Cloud Serving Benchmark (YCSB) was not designed specifically for graph databases, but rather for key-value and cloud stores. The project consists of the YCSB client which is responsible for generating the data, as well as the Core workloads those are a set of workloads executed by the client. The client is extensible to that new workloads, new databases and new generators can be integrated. [18]

The core workload is designed to use simple CRUD[4] operations on any database with no special structure of the generated data.

2.5 Other Related Work TODO: Add if more present/needed

2.5.1 Graph Database: Anna

2.5.2 TODO: Add more

[3]TODO: Enter text

[4]CRUD stands for the basic operations on persistent storage, these are Create, Read, Update, Delete.

3. Analysis

3.1 Data

3.1.1 Data StructureNOTE: Here or in Design?

3.1.2 Data Amount

3.2 Workloads

3.2.1 Inserting Data into the Database

NOTE: What is the pattern of insertion

3.2.2 Retrieving Data from the Database

NOTE: What is the pattern of retrieving

3.3 Benchmark - YCSB

TODO: show table or something to compare the different benchmarks NOTE: Activity diagram in this section. What WAS the workflow.

4. Design

4.1 Data Structure

4.2 Workloads

4.2.1 Inserting

4.2.2 Production Simulation

4.2.3 Reading under load

4.3 Extension of the Benchmark

4.3.1 Generating a Dataset

4.3.1.1 Storing the Dataset

4.3.1.2 Restoring the Dataset

4.3.2 Graph Workload

NOTE: Graph Workload functionality NOTE: Activity diagram of the workflow with graph data.

4.3.3 Bindings

4.3.3.1 Apache Jena

4.3.3.2 Neo4j

4.3.3.3 OrientDB

4.3.3.4 Sparksee

4.4 Execution Tool

4.5 Evaluation Tool

5. Implementation of your Project

5.1 Graph Data Generator

5.1.1 Parameters

5.1.2 Graph Data Creator

5.1.3 Graph Data Recreator

5.2 Graph Database Bindings

NOTE: Highlight the mapping of data and other specialities

5.2.1 Apache Jena

5.2.2 Neo4j

5.2.3 OrientDB

5.2.4 Sparksee

5.3 Graph Workload

5.3.1 Parameters

6. Evaluation

6.1 Objective

6.2 Setup

6.2.1 Hardware

6.2.2 Software

6.3 ExecutionNOTE: Scripts to run all benchmarks successively

6.4 Maximum Load

6.4.1 Probing Node CountNOTE: Comparing indexed to not indexed

6.4.1.1 Results

6.4.1.2 Discussion

6.4.2 Probing Node SizeNOTE: See change over increasing node size

6.4.2.1 Results

6.4.2.2 Discussion

6.5 Throughput

6.5.1 Difference without Edges

6.5.1.1 Results

6.5.1.2 Discussion

6.5.2 Product ComplexityNOTE: More child nodes.

6.5.2.1 Results

6.5.2.2 Discussion

6.5.3 Production SuitabilityNOTE: Testing production like workload

6.5.3.1 Results

6.5.3.2 Discussion

6.6 Responsiveness

6.6.1 Reading under load

6.6.1.1 Results

6.6.1.2 Discussion

6.6.2 Scanning under load

6.6.2.1 Results

6.6.2.2 Discussion

7. Conclusion and Future Work

7.1 Conclusion

7.1.1 Suitability

7.1.2 General Performance of Databases

7.2 Future Work

7.2.1 More Bindings

7.2.2 Concurrency

7.2.3 Other input methodsNOTE: I only used native Java APIs, to directly test the database.

7.2.4 WorkloadsTODO: what kind?

8. Summary

Bibliography

[1] Apache. *Apache Jena - Apache Jena - TDB.* http://jena.apache.org/documentation/tdb/.

[2] Apache. *Apache Jena - Fuseki serving RDF data over HTTP.* https://jena.apache.org/documen
2016.

[3] Apache. *Getting Started with Apache Jena.* https://jena.apache.org/getting_started/.
2015.

[4] Mihai Capotă et al. "Graphalytics". In: *Proc. GRADES'15 - GRADES'15* (2015), pp. 1–6. DOI: 10.1145/2764947.2764954.

[5] Miyuru Dayarathna and Toyotaro Suzumura. "XGDBench : A Bench m arking Platfor m for Graph Stores in Exascale Clouds". In: (2012), pp. 363–370.

[6] Chua Hock-Chuan. *A Quick-Start Tutorial on Relational Database Design Database Design Objectiv e.* https://www.ntu.edu.sg/home/ehchua/programming/sql/Relation

[7] Alexandru Iosup et al. *LDBC Graphalytics: A Benchmark for Large-Scale Graph Analysis on Parallel and Distributed Platforms, a Technical Report.* Tech. rep. Delft University of Technology.

[8] Myunghwan Kim and Jure Leskovec. "Multiplicative Attribute Graph Model of Real-World Networks". In: *Internet Math.* 8.1-2 (2012), pp. 113–160. ISSN: 15427951. DOI: 10.1080/15427951.2012.625257. arXiv: 1009.3499.

[9] Neo Technology Inc. *Relational Databases vs. Graph Databases: A Comparison.* https://neo4j.com/developer/graph-db-vs-rdbms/#_from_relational_to_graph_databases.
2016.

[10] Neo4j Inc. *Chapter 3. Cypher - The Neo4j Developer Manual v3.3.* https://neo4j.com/docs/deve
manual/current/introduction/#introduction-highlights.

[11] Ontotext. *The Truth About Triplestores.* https://ontotext.com/wp-content/uploads/2014/07/T
Truth-About-Triplestores.pdf. 2014.

[12] Orient Technologies. *Getting Started · OrientDB Manual.* https://orientdb.com/docs/last/Tuto
Introduction-to-the-NoSQL-world.html.

[13] Margaret Rouse. *What is data collection? - Definition from WhatIs.com.* https://whatis.techtarg
database. 2016.

[14] Sparsity Technologies. *User Manual - Sparksee.* Tech. rep., p. 147.

[15] Techopedia. *What is a Data Dictionary? - Definition from Techopedia.* https://www.techopedia.
oriented-database. 2017.

[16] W3C. *Semantic Web Standards.* https://www.w3.org/RDF/. 2014.

[17] Thomas Worsch. "Grundbegriffe der Informatik". In: November (2011).

[18] Yahoo! *Yahoo! Cloud Serving Benchmark (YCSB) Wiki.* https://research.yahoo.com/news/yaho
cloud-serving-benchmark/?guccounter=1. 2010.

[19] Serdar Yegulalp. *What is NoSQL? NoSQL databases explained | InfoWorld.* https://www.infoworld.com/article/3240644/nosql/what-is-nosql-nosql-databases-explained.html. 2017.