KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in ?

Second cycle, 30 credits

Delta Lake as Offline Feature Store for Hopsworks

A subtitle in the language of the thesis

GIOVANNI MANFREDI

Stockholm, Sweden, 2024

Delta Lake as Offline Feature Store for Hopsworks

A subtitle in the language of the thesis

GIOVANNI MANFREDI

Master's Programme, ICT Innovation, 120 credits
Date: March 1, 2024

Supervisors: Sina Sheikholeslami, Fabian Schmidt
Examiner: Vladimir Vlassov
School of Electrical Engineering and Computer Science
Host company: Hopsworks AB
Swedish title: Detta är den svenska översättningen av titeln
Swedish subtitle: Detta är den svenska översättningen av undertiteln

Abstract

Here I will write an abstract that is about 250 and 350 words (1/2 A4-page) with the following components:

- What is the topic area? (optional) Introduces the subject area for the project.
- Short problem statement
- Why was this problem worth a Bachelor's/Master's thesis project? (*i.e.*, why is the problem both significant and of a suitable degree of difficulty for a Bachelor's/Master's thesis project? Why has no one else solved it yet?)
- How did you solve the problem? What was your method/insight?
- Results/Conclusions/Consequences/Impact: What are your key results/ conclusions? What will others do based on your results? What can be done now that you have finished - that could not be done before your thesis project was completed?

Keywords

Canvas Learning Management System, Docker containers, Performance tuning First keyword, Second keyword, Third keyword, Fourth keyword

Sammanfattning

Här ska jag skriva ett abstract som är på ca 250 och 350 ord (1/2 A4-sida) med följande komponenter:

- Vad är ämnesområdet? (valfritt) Presenterar ämnesområdet för projektet.
- Kort problemformulering
- Varför var detta problem värt en kandidat-/masteruppsats? (*i.e.*, varför är problemet både betydande och av en lämplig svårighetsgrad för ett kandidat-/masteruppsats-projekt? Varför har ingen annan löst det än?)
- Hur löste du problemet? Vad var din metod/insikt?
- Resultat/slutsatser/konsekvenser/påverkan: Vilka är dina viktigaste resultat/
  slutsatser? Vad kommer andra att göra baserat på dina resultat? Vad kan göras nu när du är klar - som inte kunde göras innan ditt examensarbete var klart?

Nyckelord

Canvas Lärplattform, Dockerbehållare, Prestandajustering Första nyckelordet, Andra nyckelordet, Tredje nyckelordet, Fjärde nyckelordet

Sommario

Qui scriverò un abstract di circa 250 e 350 parole (1/2 pagina A4) con i seguenti elementi:

- Qual è l'area tematica? (opzionale) Introduce l'area tematica del progetto.
- Breve esposizione del problema
- Perché questo problema meritava un progetto di tesi di laurea/master? (Perché il problema è significativo e di un grado di difficoltà adeguato per un progetto di tesi di laurea/master? Perché nessun altro l'ha ancora risolto?)
- Come avete risolto il problema? Qual è stato il vostro metodo/intuizione?
- Risultati/Conclusioni/Conseguenze/Impatto: Quali sono i vostri risultati chiave/ conclusioni? Cosa faranno gli altri sulla base dei vostri risultati? Cosa si può fare ora che avete finito - che non si poteva fare prima che il vostro progetto di tesi fosse completato?

parole chiave

Prima parola chiave, Seconda parola chiave, Terza parola chiave, Quarta parola chiave

Acknowledgments

I would like to thank xxxx for having yyyy.

Stockholm, June 2024
Giovanni Manfredi

Contents

List of acronyms and abbreviations

ACID	Atomicity, Consistency, Isolation and Durability
AI	Artificial Intelligence
BI	Business Intelligence
ELT	Extract Load Transform
ETL	Extract Transform Load
ML	Machine Learning
OLAP	On-Line Analytical Processing
RDD	Resilient Distributed Dataset

Chapter 1

Introduction

Lakehouse systems are increasingly becoming the primary choice for running analytics in large sized companies (that have more than 1000 employees) [1].

This recent architecture design [2] is preferred over old paradigms, i.e. data warehouses and data lakes, because it takes the best of both worlds, having scalability properties of data lakes, while preserving the Atomicity, Consistency, Isolation and Durability (ACID) properties typical of data warehouses. Additionally, Lakehouse systems include partitioning, that reduces query significantly and time travel enabling users to access to different versions of data, versioned over time [3].

Three implementations of this paradigm emerged over time [4]:

1. **Apache Hudi**: first introduced by Uber, now primarily backed by Uber, Tencent, Alibaba and Bytedance
2. **Apache Iceberg**: first introduced by Netflix and now primarily backed by Netflix, Apple and Tencent
3. **Delta Lake**: first introduced by Databricks and now primarily backed by Databricks and Microsoft

While all three projects are backed by large communities, it is Delta Lake that is more acknowledged as Lakehouse solution [4]. This is mainly thanks to Databricks, that first promoted this new architecture over data lakes among their clients around 2020 [5].

Delta Lake is typically used in combination with Apache Spark [6] that acts as data query and processing engine. This approach is effective when processing large quantities of data (1 TB or more) over the cloud, but is this approach still effective in other scenarios such as local computing over small quantities of data (100 GB or less)?

In recent years, in other data storage cases, DuckDB [7] and Polars [8], showed as under a certain number of data volume, and in particular if the architecture is local, starting a Spark cluster, might actually reduce performance, increasing costs and the computation time. Spark alternatives at smaller scale (10 GB - 100 GB) generally perform much better in these scenarios [9, 10].

Another aspect to keep in mind when developing in this field is which programming language data scientist like to use, and this is Python. Python is the currently the most popular programming language [11] and it is by far the most used language for Machine Learning (ML) and Artificial Intelligence (AI) applications [12], this is mainly thanks to its strong abstraction capabilities and accessibility. This can be also observed looking at the mentioned libraries, DuckDB, Polars and Spark, that all offer Python interfaces. In this scenario, creating a Python client for Delta Lake would be beneficial as it would not have to resort to Spark and its Python library (PySpark). This approach would with small scale use cases would improve performance significantly.

This native Python interface for Delta Lake directly benefits Hopsworks AB, the host company of this master thesis. Hopsworks develops a ML platform that enables developers to build, maintain and monitor ML systems [**?**]. This implementation would be integrated in their ML platform enabling developers using it, a faster access to Delta Lake compared to Spark at a smaller data scale (typically between 10 GB and 100 GB).

1.1 Background

A cleared understanding of the background of this project comes from appreciating three different key aspects: Lakehouse development, Spark relevance and flows, Python as emergent language.

Lakehouse is a term coined by Databricks in 2020 [2], to define a new design standard that was emerging in the industry, that combined the capability of data lakes of storing and managing unstructured data, with the ACID properties typical of Data warehouses. Data warehouses became a dominant standard in the 90s early 2000s, enabling companies to generate Business Intelligence (BI) insights, managing different structured data sources. The problems related to this architecture rose in the 2010 years when it became clear the need to manage unstructured data in large quantities [13]. So Data lakes became the pool where all data could be stored, on top of which a more complex architecture could be built, consisting of

data warehouses for BI and ML pipelines. This architecture, while more suitable for unstructured data, introduces many complexities and costs, related to the need of having replicated data (data lake and data warehouse), and a lot of Extract Load Transform (ELT) and Extract Transform Load (ETL) computations. Lakehouses solved the problems of Data lakes by implementing data management and performance features on top of open data formats such as Parquet [14]. This paradigm was successful thanks to three key components: a metadata layer for data lakes, a new query engine design, a declarative access for ML and AI. This architecture design was first open-sourced with Apache Hudi in 2017 [15] and then Delta Lake in 2020 [5].

When talking about Spark origins and reasons behind its creation we need to look into Google needs for advancing in the internet search and indexing. In particular these needs led to the creation of MapReduce [16] a distributed programming model that enables the management of large datasets. This paradigm became then part of the Hadoop ecosystem [17]. From the roots of MapReduce, Spark was created [6], improving both on the performance (10 times better in its first iteration), and on the fault tolerance, using Resilient Distributed Datasets (RDDs). RDDs are a distributed memory abstraction that enables a lazy in-memory computation that is tracked though the use of lineage graphs, ultimately increasing fault tolerance [18]. Spark, now Apache Spark under the Apache foundation [19], has seen widespread success and adoption in various applications, becoming a de-facto standard of the distributed computing world. As Spark becomes older, other approaches appear and compete with Spark in specific areas, achieving better results. This is the case of Apache Flink [20], designed for true stream processing prevails over Spark in this area. The same can be said for lower scale applications where the high scaling capabilities of Spark are not at use, and the overhead of starting a Spark application is not compensated by other factors. This is the case of DuckDB [7] and Polars [8], that focusing on low scale (10GB-100GB) they provided a fast On-Line Analytical Processing (OLAP) embedded database and DataFrame management system respectively offering an overall faster computation compared to starting a Spark cluster for to perform the same operations. This shows the possibility for improvements and new applications that substitute the current Spark-based systems in specific applications.

The data science world speaks Python [21]. Python was first adopted by many thanks to its focus on ease of use, high abstraction level and readability. This helped created a fast-growing community behind the project, that lead to the development of a great number of libraries and APIs. So now, after more than 30 years after its creation, it became the de-facto standard for data

science thanks to its many libraries such as Tensorflow, NumPy, SciPy, Pandas, PyTorch, Keras and many others.

Python is also the most popular programming language. This appears clear if we refer to TIOBE Index 2024 [11] we see that Python has a rating of 15.16%, followed by C that has a rating of 10.97%. The index also shows the trends of the last years, clearly displaying the rise of Python over historically very popular languages such as C and JAVA, that were both outranked by Python between 2021 and 2022. This shows the importance of offering Python interfaces for programmers and data scientist in particular to increase the engagement and possibilities of a framework.

1.2 Problem

Longer problem statement
If possible, end this section with a question as a problem statement.

1.2.1 Original problem and definition

Some text

1.2.2 Scientific and engineering issues

some more text

1.3 Purpose

In this section I will state the purpose of your thesis and the purpose of your degree project.
Describe who benefits and how they benefit if you achieve your goals. Include anticipated ethical, sustainability, social issues, etc. related to your project.

1.4 Goals

The goal of this project is XXX. This has been divided into the following three sub-goals:

1. Subgoal 1
2. Subgoal 2

3. Subgoal 3

1.5 Research Methodology

In this section I will present your philosophical assumption(s), research method(s), and research approach(es).

1.6 Delimitations

In this section I will describe the boundary/limits of your thesis project and what you are explicitly not going to do. This will help you bound your efforts - as you have clearly defined what is out of the scope of this thesis project. Explain the delimitations. These are all the things that could affect the study if they were examined and included in the degree project.

1.7 Structure of the thesis

This section will describe how the thesis is structured indicating various chapters.

References

[1] "State of the Data Lakehouse," Dremio, Tech. Rep., 2024. [Page 1.]

[2] M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, "Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics," in *Proceedings of CIDR*, vol. 8, 2021. [Pages 1 and 2.]

[3] D. Croci, "Data Lakehouse, beyond the hype," Dec. 2022. [Page 1.]

[4] "Apache Hudi vs Delta Lake vs Apache Iceberg - Data Lakehouse Feature Comparison," https://www.onehouse.ai/blog/apache-hudi-vs-delta-lake-vs-apache-iceberg-lakehouse-feature-comparison. [Page 1.]

[5] M. Armbrust, T. Das, L. Sun, B. Yavuz, S. Zhu, M. Murthy, J. Torres, H. Van Hovell, A. Ionescu, A. Łuszczak, M. Świtakowski, M. Szafrański, X. Li, T. Ueshin, M. Mokhtar, P. Boncz, A. Ghodsi, S. Paranjpye, P. Senster, R. Xin, and M. Zaharia, "Delta lake: High-performance ACID table storage over cloud object stores," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3411–3424, Aug. 2020. doi: 10.14778/3415478.3415560 [Pages 1 and 3.]

[6] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache Spark: A unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, Oct. 2016. doi: 10.1145/2934664 [Pages 1 and 3.]

[7] M. Raasveldt and H. Mühleisen, "DuckDB: An Embeddable Analytical Database," in *Proceedings of the 2019 International Conference on Management of Data*. Amsterdam Netherlands: ACM, Jun. 2019. doi: 10.1145/3299869.3320212. ISBN 978-1-4503-5643-5 pp. 1981–1984. [Pages 2 and 3.]

[8] R. Vink, "I wrote one of the fastest DataFrame libraries," https://www.ritchievink.com/blog/2021/02/28/i-wrote-one-of-the-fastest-dataframe-libraries/, Feb. 2021. [Pages 2 and 3.]

[9] "Benchmark Results for Spark, Dask, DuckDB, and Polars — TPC-H Benchmarks at Scale," https://tpch.coiled.io/. [Page 2.]

[10] T. Ebergen, "Updates to the H2O.ai db-benchmark!" https://duckdb.org/2023/11/03/db-benchmark-update.html, Nov. 2023. [Page 2.]

[11] "TIOBE Index," https://www.tiobe.com/tiobe-index/. [Pages 2 and 4.]

[12] S. Raschka, Vahid and Mirjalili, *Python Machine Learning (3rd Edition)*. Packt Publishing, 2019. ISBN 978-1-78995-575-0 [Page 2.]

[13] EDER, "Unstructured Data and the 80 Percent Rule," Aug. 2008. [Page 2.]

[14] "Dremel made simple with Parquet," https://blog.x.com/engineering/en_us/a/2013/dremel-made-simple-with-parquet. [Page 3.]

[15] P. Rajaperumal, "Uber Engineering's Incremental Processing Framework on Hadoop," https://www.uber.com/blog/hoodie/, Mar. 2017. [Page 3.]

[16] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008. doi: 10.1145/1327452.1327492 [Page 3.]

[17] "Apache Hadoop," https://hadoop.apache.org/. [Page 3.]

[18] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2011-82, Jul. 2011. [Page 3.]

[19] "Apache Spark™ - Unified Engine for large-scale data analytics," https://spark.apache.org/. [Page 3.]

[20] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, “Apache Flink™: Stream and Batch Processing in a Single Engine.” [Page 3.]

[21] G. van Rossum, “Python tutorial,” Centrum voor Wiskunde en Informatica (CWI), Amsterdam, Tech. Rep. CS-R9526, May 1995. [Page 3.]