Key-Value vs Graph-based data lakes for realizing Digital Twin systems (Poster)

Daniel Pérez-Porras, Paula Muñoz, Javier Troya and Antonio Vallecillo

ITIS Software. University of Málaga, Spain

Keywords
Digital twins, Data Lake, NoSQL databases, Graph databases

1. Introduction

A Digital Twin (DT) is a comprehensive digital representation of an actual system, service or product (the Physical Twin, PT), synchronized at a specified frequency and fidelity [1]. The digital twin includes the properties, condition and behavior of the physical entity through models and data, and is continuously updated with real-time data about the PT performance, maintenance, and health status throughout its entire lifetime [2]. The exchange of data between the digital and the physical twins takes place through bi-directional data connections. Additionally, a DT system can also comprise a set of services that permit exploiting the data exchanged by the two twins [3].

Engineering DT systems is challenging for many reasons, one of them being their complexity [4]. The problem we would like to address in this paper is how to implement the connections between the twins in an effective and efficient way. Usually, these connections are achieved through a Data Lake. As defined in [5], a data lake is "a flexible, scalable data storage and management system, which ingests and stores raw data from heterogeneous sources in their original format, and provides query processing and data analytics in an on-the-fly manner."

In a previous work [6] we defined a framework for the specification and deployment of DT systems. It uses UML models to specify the digital twins, and connects them through a Data Lake repository implemented in Redis (https://redis.io/), which provides the bi-directional communication infrastructure. This open-source lightweight in-memory data structure is optimized to deliver fast responses to a massive amount of petitions. Redis is a key-value database and supports various abstract data structures such as strings, lists, sets or maps (called 'hashes'). However, Redis does not easily allow complex queries: to retrieve hashes by the values of their fields, it is necessary to store additional records that include the field value and a reference to the hash key. This makes the database structure and contents dependent on the queries that need to be performed.

MESS@STAF 2022: International workshop on MDE for Smart IoT Systems, July 04–08, 2022, Nantes, France

daniperezporras@uma.es (D. Pérez-Porras); paulam@uma.es (P. Muñoz); jtroya@uma.es (J. Troya); av@uma.es (A. Vallecillo)
0000-0003-2939-5803 (P. Muñoz); 0000-0002-1314-9694 (J. Troya); 0000-0002-8139-9986 (A. Vallecillo)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
CEUR Workshop Proceedings (CEUR-WS.org)

CEUR-WS.org/Vol-3250/messpaper2.pdf

Alternative solutions use time series databases, such as InfluxDB or TimescaleDB, and even define temporal models for handing them at a higher level of abstraction [7]. While very efficient for querying time-sensitive information, these solutions are not optimal for implementing more general queries, needed when efficient data analysis is required.

In this work we explore the use of Graph databases [8] to store and query the information handled in data lakes. Graph databases use graph structures to perform semantic queries; they store the information as nodes, edges, and properties. Examples of Graph databases include Neo4j, ArangoDB or OrientDB, to name a few. They all count with specialized query languages such as Gremlin, Cypher, SPARQL, or GraphQL.

We have implemented a data lake using Neo4j, and evaluated its performance and expressiveness against our previous implementation with Redis. As expected, Neo4j allows easy specification of more complex queries using its Cypher query language. The same queries in Redis require the addition of ad-hoc records with the corresponding key-value mappings if the queries were not contemplated beforehand in the Redis record structure. Interestingly, the additional queries forced by these new records introduce a performance penalty that makes Neo4j's response times better than those of Redis. Furthermore, the response times obtained for simple queries in Redis are not very different from those of Neo4j. All this makes Neo4j appear to be a better solution than Redis for realizing data lakes. The description of the tests carried out and their results are availble from https://github.com/atenearesearchgroup/dt-graph-database. As future work, we are defining a benchmark with different types of queries that will be used to compare implementations of data lakes using different technologies, including time series databases, too. We hope that our evaluation will help to shed some light on the advantages and limitations of each solution, and to identify situations where one type of solution outperforms the other.

References

[1] Digital Twin Consortium, Glossary of digital twins, https://www.digitaltwinconsortium.org/glossary/index.htm, 2021.

[2] F. Bordeleau, B. Combemale, R. Eramo, M. van den Brand, M. Wimmer, Towards model-driven digital twin engineering: Current opportunities and future challenges, in: Proc. of ICSMM'20, volume 1262 of CCIS, Springer, 2020, pp. 43–54.

[3] F. Tao, H. Zhang, A. Liu, A. Y. C. Nee, Digital twin in industry: State-of-the-art, IEEE Trans. Ind. Informatics 15 (2019) 2405–2415. doi:10.1109/TII.2018.2873186.

[4] M. Grieves, J. Vickers, Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems, Springer, 2017, pp. 85–113.

[5] R. Hai, C. Quix, M. Jarke, Data lake concept and systems: a survey, CoRR abs/2106.09592 (2021). arXiv:2106.09592.

[6] P. Muñoz, J. Troya, A. Vallecillo, Using UML and OCL Models to Realize High-Level Digital Twins, in: Proc. of ModDiT2021@MODELS'21, IEEE, 2021, pp. 212–220.

[7] A. Mazak, S. Wolny, A. Gómez, J. Cabot, M. Wimmer, G. Kappel, Temporal models on time series databases, J. Object Technol. 19 (2020) 3:1–15.

[8] I. Robinson, J. Webber, E. Eifrem, Graph Databases: New Opportunities for Connected Data, 2 ed., O'Reilly Media, Inc., 2015.