Graph-Inceptor: Towards Extreme Data Ingestion, Massive Graph Creation and Storage

Jože Rožanec
Jožef Stefan Institute
Slovenia
joze.rozanec@ijs.si

Brian Elvesæter
SINTEF
Norway
brian.elvesater@sintef.no

Dumitru Roman
SINTEF
Norway
dumitru.roman@sintef.no

Marko Grobelnik
Jožef Stefan Institute
Slovenia
marko.grobelnik@ijs.si

Peter Haase
Metaphacts
Germany
ph@metaphacts.com

ABSTRACT

Graph processing is increasingly popular given the wide range of phenomena represented as graphs (e.g., social media networks, pharmaceutical drug compounds, or fraud networks, among others). The increasing amount of data available requires new approaches to efficiently ingest and process such data. In this research, we describe a solution at a conceptual level in the context of the Graph-Massivizer architecture. Graph-Inceptor aims to bridge the void among ETL tools enabling data transformations required for graph creation and enrichment and supporting connectors to multiple graph storages at a massive scale. Furthermore, it aims to enhance ETL operations by learning from data content and load and making decisions based on machine-learning-based predictive analytics.

CCS CONCEPTS

• **Computer systems organization → Architectures**.

ACM Reference Format:
Jože Rožanec, Brian Elvesæter, Dumitru Roman, Marko Grobelnik, and Peter Haase. 2023. Graph-Inceptor: Towards Extreme Data Ingestion, Massive Graph Creation and Storage. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3578245.3585339

1 INTRODUCTION

Knowledge graphs represent semantically structured information and offer great potential for building more intelligence into existing software systems [24]. The increasing amount of available data makes it increasingly challenging to construct, and process graphs using a single machine [11]. Furthermore, it is becoming essential to concurrently ingest fine-grained updates at high velocities and a massive scale while supporting efficient data access for multiple purposes [12, 17]. A scalable framework for graph Extract-Transform-Load (ETL) must consider the tasks related to graph formation and serialization. RDF has been widely used to represent graphs, and many ingestion and data processing tools support it [16]. Furthermore, specialized graph serialization formats have been developed to guarantee space-efficient graph representations. Among them, we can mention the Yale format (representing a graph using three arrays, accounting for non-zero values, the extent of rows, and column indices); WebGraph (optimized for representing a web graph, exploiting links locality) [4]; or the K^2-tree [5].

Distributed graph persistence and processing have been addressed in many ways. Since Google introduced Pregel [14], many graph processing frameworks with diverse programming models and features have been proposed, usually persisting graph data across a distributed filesystem [8]. On the other hand, specialized databases have been developed to support distributed storage (e.g., Titan, OrientDB, ArangoDB) and complex queries on top of them. Distributed graphs can be either persisted and processed in distributed homogeneous or heterogeneous architectures [3, 10]. Heterogeneous architectures pose additional challenges to exploiting the capabilities of each node best. Graph partitioning must be considered for graphs persisted in a distributed manner, given the direct impact on the performance imposed by load balancing and communication traffic between the instances of a cluster [6].

Graph neural networks have demonstrated ground-breaking performance on many tasks [20, 22]. While data growth has pushed towards a distributed processing paradigm, research on graph neural networks has been mainly performed considering shared-memory architectures within a single machine. Efforts to enable distributed graph neural networks have been few and recent [9, 13, 21].

Many ETL products (e.g., Airbyte, Stitch, Fivetran, and many others) provide generic connectors to load and persist tabular data and neglect connectors to graph storage. Frameworks such as the data build tool (DBT) [7] capture ETL best practices; however, they are limited to SQL query language for data transformations. Therefore, there is a need to develop tools that support ingesting massive amounts of data, allowing for fine-grained updates at high velocities, and storing the data in multiple target stores to satisfy heterogeneous processing needs.

In the context of the Graph-Massivizer project and proposed architecture [15], we propose developing the Graph-Inceptor tool. We describe the tool in detail in the next section.

Jože M. Rožanec and Brian Elvesæter are co-first authors with equal contribution and importance.

ICPE '23 Companion, April 15–19, 2023, Coimbra, Portugal

ACM ISBN 979-8-4007-0072-9/23/04.
https://doi.org/10.1145/3578245.3585339

2 PROPOSED ARCHITECTURE

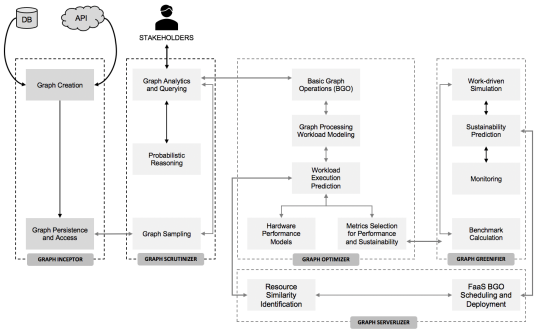

Figure 1: The Graph-Massivizer technical architecture consists of five tools.

Graph-Inceptor aims to enable scalable data ingestion and graph creation. Developed as part of the Graph-Massivizer architecture (see Fig. 1), it was inspired by the AliGraph architecture [23], which considers a distributed graph storage with multiple partitions, separate storage for attributes, and a cache for most essential vertices. We consider the Graph-Inceptor requires two clearly defined components: (a) an Extraction-Transform (ET) process and (b) a Load (L) process. ET is concerned with extracting data from external data sources (e.g., databases, APIs, and knowledge graphs) and transforming it (e.g., extracting entities and relationships from text, creating semantic abstractions) to be later inserted into the graph. On the other side, the L provides means to persist data that was loaded and transformed. The L process may include additional steps geared towards alleviating graph post-processing, e.g., by performing graph sampling in an online manner [1, 2].

We envision the whole ETL process can be enhanced with artificial intelligence models applied across the many phases of the ingestion process. Such enhancement can be performed at least in two dimensions: (a) to better process the incoming data and ensure a higher quality of results and (b) to optimize the underlying infrastructure. An example of enhancing the data processing could be using natural language processing and graph neural networks to extract relevant information from the text and embed it into the graph. Furthermore, classifiers could be used to determine associated sentiments, topics, and other relevant aspects of interest. Conversely, machine learning could optimize an underlying infrastructure, e.g., by learning the best cache replacement policies [19] or predicting the best graph partitioning strategies [18].

ACKNOWLEDGMENT

The Graph-Massivizer project receives funding from the Horizon Europe research and innovation program of the European Union under grant agreement 101093202. The project started on January 1st 2023 and lasts three years.

REFERENCES

[1] Nesreen K Ahmed, Nick Duffield, Jennifer Neville, and Ramana Kompella. 2014. Graph sample and hold: A framework for big-graph analytics. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1446–1455.

[2] Nesreen K Ahmed, Nick Duffield, Theodore Willke, and Ryan A Rossi. 2017. On sampling from massive graph streams. *arXiv preprint arXiv:1703.02625* (2017).

[3] Maciej Besta, Dimitri Stanojevic, Johannes De Fine Licht, Tal Ben-Nun, and Torsten Hoefler. 2019. Graph processing on fpgas: Taxonomy, survey, challenges. *arXiv preprint arXiv:1903.06697* (2019).

[4] P. Boldi and S. Vigna. 2004. The Webgraph Framework I: Compression Techniques. In *Proceedings of the 13th International Conference on World Wide Web* (New York, NY, USA) *(WWW '04)*. Association for Computing Machinery, New York, NY, USA, 595–602. https://doi.org/10.1145/988672.988752

[5] Nieves R Brisaboa, Susana Ladra, and Gonzalo Navarro. 2014. Compact representation of web graphs with extended functionality. *Information Systems* 39 (2014), 152–174.

[6] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. 2016. *Recent advances in graph partitioning*. Springer.

[7] dbt (data build tool. 2023. dbt - transform data in your warehouse. https://www.getdbt.com/

[8] Arturo Diaz-Perez, Alberto Garcia-Robledo, and Jose-Luis Gonzalez-Compean. 2019. *Graph Processing Frameworks*. Springer International Publishing, Cham, 875–883. https://doi.org/10.1007/978-3-319-77525-8_283

[9] Swapnil Gandhi and Anand Padmanabha Iyer. 2021. P3: Distributed Deep Graph Learning at Scale.. In *OSDI*. 551–568.

[10] Safiollah Heidari, Yogesh Simmhan, Rodrigo N Calheiros, and Rajkumar Buyya. 2018. Scalable graph processing frameworks: A taxonomy and open challenges. *ACM Computing Surveys (CSUR)* 51, 3 (2018), 1–53.

[11] Nilesh Jain, Guangdeng Liao, and Theodore L Willke. 2013. Graphbuilder: scalable graph etl framework. In *First international workshop on graph data management experiences and systems*. 1–6.

[12] Pradeep Kumar and H Howie Huang. 2020. Graphone: A data store for real-time analytics on evolving graphs. *ACM Transactions on Storage (TOS)* 15, 4 (2020), 1–40.

[13] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. 2019. Pytorch-biggraph: A large scale graph embedding system. *Proceedings of Machine Learning and Systems* 1 (2019), 120–131.

[14] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. 2010. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 135–146.

[15] Radu Prodan, Dragi Kimovski, Andrea Bartolini, Michael Cochez, Alexandru Iosup, Evgeny Kharlamov, Jože Rožanec, Laurenţiu Vasiliu, and Ana Lucia Vărbănescu. 2022. Towards Extreme and Sustainable Graph Processing for Urgent Societal Challenges in Europe. In *2022 IEEE Cloud Summit*. IEEE, 23–30. https://doi.org/10.1109/CloudSummit54781.2022.00010

[16] Dumitru Roman, Nikolay Nikolov, Antoine Putlier, Dina Sukhobok, Brian Elvesæter, Arne Berre, Xianglin Ye, Marin Dimitrov, Alex Simov, Momchill Zarev, et al. 2018. DataGraft: One-stop-shop for open data management. *Semantic Web* 9, 4 (2018), 393–411. https://doi.org/10.3233/SW-170263

[17] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid Aref, Marcelo Arenas, Maciej Besta, Peter A Boncz, et al. 2021. The future is big graphs: a community view on graph processing systems. *Commun. ACM* 64, 9 (2021), 62–71. https://doi.org/10.1145/3434642

[18] Jiayi Shen and Fabrice Huet. 2018. Predict the best graph partitioning strategy by using machine learning technology. In *Proceedings of the 2018 VII International Conference on Network, Communication and Computing*. 27–33.

[19] Giuseppe Vietri, Liana V Rodriguez, Wendy A Martinez, Steven Lyons, Jason Liu, Raju Rangaswami, Ming Zhao, and Giri Narasimhan. 2018. Driving Cache Replacement with ML-based LeCaR.. In *HotStorage*. 928–936.

[20] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[21] Da Zheng, Chao Ma, Minjie Wang, Jinjing Zhou, Qidong Su, Xiang Song, Quan Gan, Zheng Zhang, and George Karypis. 2020. Distdgl: distributed graph neural network training for billion-scale graphs. In *2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3)*. IEEE, 36–44.

[22] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI open* 1 (2020), 57–81.

[23] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. 2019. Aligraph: A comprehensive graph neural network platform. *arXiv preprint arXiv:1902.08730* (2019).

[24] Xiaohan Zou. 2020. A survey on application of knowledge graph. In *Journal of Physics: Conference Series*, Vol. 1487. IOP Publishing, 012016.