Optimizing Data Integration Processes with the Support of Machine Learning - Is it Really Possible?

Robert Wrembel

Poznan University of Technology and
Interdisciplinary Centre for Artificial Intelligence and Cybersecurity, pl. Sklodowskiej-Curie 5, 60965, Poznań, Poland

Abstract

In this panel session I address two research questions in the area of the optimization of data integration (DI) processes (a.k.a. ETL processes), which (in my opinion) still need substantial research. The questions include: (1) how to efficiently push down executions of DI tasks to non-relational data sources and (2) how to handle user-defined functions (especially treated as black-boxes) in optimizing the performance of DI processes. The discussion to be initiated during the panel is whether sound answers to these questions can be found by the support of machine learning techniques.

Keywords

data integration process, ETL process, optimizing data integration process, user-defined functions, resource usage time series, machine learning, time series similarity

1. Data integration architectures and processes

For years, the widespread of complex, data-driven systems has been observed, e.g., medical systems, smart agriculture, and smart cities. These systems produce huge volumes of highly heterogeneous data (a.k.a. big data) that need to be integrated to feed various applications providing descriptive analytics or prediction models. Thus, data integration (DI) architectures are inevitable in modern information systems and they are constantly facing new challenges caused by complex, fast arriving, and ample data as well as emerging data engineering technologies.

A common goal of DI is to make heterogeneous and typically distributed data available for an end user in a unified format. Research and development works resulted in a few standard DI architectures, namely: (1) federated [1] and mediated [2], (2) data warehouse (DW) [3], (3) lambda [4], (4) data lake (DL) [5], (5) data lake house (DLH) [6], (6) polystore [7], and (7) data mesh/ data fabric [8]. In all of the aforementioned architectures, data from heterogeneous and distributed data sources (DSs) are made available in an integrated system (either by virtual or materialized integration) by means of an integration layer. This layer is implemented by a sophisticated software, which runs the so-called DI processes (a.k.a. ETL - in data warehouse architectures, data processing pipeline - in data science, data wrangling, or data processing workflows [9, 10]).

DI processes are core elements of all DI architectures. DI processes are complex workflows composed of dozens to thousands of tasks. These tasks are responsible for extracting data from DSs, transforming data into a common model and data structures, cleaning data, removing missing, inconsistent, and redundant data items, integrating data, and loading them into a central repository (i.e., DW, DL, or DLH) or making them available in virtual integration architectures (i.e., federated, mediated, polystore, or data mesh). DI processes are managed by a dedicated software, called a DI engine (an ETL engine in a DW architecture). Most of the DI engines support a set of predefined (out of the box) tasks [11].

Even though, methods for developing DI processes have been researched and developed for decades (see [10, 12]) and were included in commercial (and some open license) DI design environments and DI engines [13], the task of designing and managing DI processes is still difficult and time costly. Moreover, the support from these design tools for optimizing such designs and **optimizing the execution of DI processes** is very limited.

In this context, with the fast advances of machine learning (ML) techniques, the application of such techniques to designing and optimizing DI processes may sound attractive. However, research works on DI focus mainly on mappings between values [14] or schemas [15], data cleaning [16], data deduplication [17, 18]. Moreover, event though multiple providers of DI technologies and consulting companies opt for applying ML techniques in data integration, a clear step-by-step and end-to-end approach has not been proposed yet.

ML techniques have already been successfully applied to optimizing system performance, e.g., [19, 20, 21, 22, 23, 24, 25]. They typically build performance models, which are based on performance characteristics (typically CPU, I/O, and memory usage) collected during a normal runtime of a system or during an excessive testing phases. Then, performance models are learned, based on these characteristics. The works reported in [26, 27] focus on applying ML techniques to provide auto-tuning capabilities in the so-called self-driving database management systems.

In this panel talk, I will focus on selected challenges related to the performance of ETL processes. My subjective point of view on the presented open issues/challenges results from a cooperation with IBM Software Lab in Kraków (Poland) on a data integration project.

2. Performance optimization of DI processes

In order to reduce the execution time of a DI process, a few classes of solutions have been proposed. First, a business approach is to scale-up or scale-out a DI server. Second, DI engines existing on the market support parallel processing of DI tasks. This is also a trend in research. Third, some

DOLAP 2024: 26th International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data, co-located with EDBT/ICDT 2024, March 25, 2024, Paestum, Italy

robert.wrembel@put.poznan.pl (R. Wrembel)
http://www.cs.put.poznan.pl/rwrembel/ (R. Wrembel)
0000-0001-6037-5718 (R. Wrembel)

DI engines support moving the execution of some DI tasks close to storage. One technique of this class is called the push-down optimization. Fourth, re-ordering of DI tasks has been well researched and resulted in a few approaches.

Scaling refers to adding computing power into a DI architecture. Two types of scaling are common, namely: (1) vertical scaling of a DI server, i.e., by increasing the number of CPUs, the size of RAM, adding specialized hardware like FPGAs and (2) horizontal scaling of a DI architecture, i.e., adding new computing nodes.

Parallel processing consists in computing tasks by parallel OS processes or threads. This technique was well researched as well, e.g., [28, 29, 30]. In the simplest case (available in commercial DI engines, see [13]), uploading data into a data warehouse is executed in parallel. A challenge in applying parallelism is to figure out the most efficient parallelization schemes for a given DI task or the whole DI process [31].

Data processing close to storage - in the context of databases, there are numerous implementations supporting moving data-intensive processing from an application layer to storage. Examples of such systems are IBM Pure Data for Analytics and Oracle Exadata. Both of them use a dedicated hardware to perform operations on data read from disks, like decompression, filtering, and projection.

Push-down - the principle of the *push-down* optimization is to move some DI tasks into a data source, to be executed there. Push-down is available in IBM InfoSphere Data Stage and Informatica, but only for relational DSs.

Task reordering is the most researched technique for DI process optimization. A group of approaches draws upon the idea of changing the order of tasks in an original DI process, such that a reordered process is more efficient than the original one. Finding a (sub-)optimal order of tasks is computationally complex, and for this reason, some heuristics have to be used [32, 33].

3. Still open research challenges

This section outlines my subjective view on two still open research challenges in the optimization of DI processes. They include: the push-down optimization on non-relational data sources and DI processes with black-box user-defined functions (BBUDFs).

As stated above, the **push-down** technique in commercial systems was made available to work only with relational DSs and it is typically applicable to tasks at the beginning of a DI process, i.e., filtering or simple pre-processing. Moreover, push-down may also be applicable to tasks that migrate large volumes of data between systems. The first example of such a task is data anonymization. According to GDPR, sensitive data cannot 'leave' a source system before being anonymized. It means that a DI engine cannot run the anonymization and this task has to be pushed-down into the source system. The second example is enforcing data access policies. Sensitive data that cannot be accessed in the source system must be filtered out directly in the system. In this case, a data access policy originally included in a DI process must be pushed-down into the DS.

With the widespread of big data storage systems, a natural step is to extend push-down to non-relational DSs. To the best of our knowledge, the applicability and efficiency of this technique for non-relational (a.k.a. NoSQL) DSs has not been studied yet (with the exception of [34, 35, 36]). The issues that have to be investigated include: (1) analyzing which DI tasks can be pushed down to contribute to the improvement of performance of a DI process and (2) how to efficiently implement a given pushed down task in a DS, leveraging the functionality and internal structures of the DS.

DI processes use not only predefined tasks (e.g., [11]) available in design, development, and management tools [13], but also require the deployment of **user defined functions** (UDFs), in order to implement specific tasks [37, 38]. UDFs can be implemented in various programming languages and are treated by a DI engine as **black-boxes**. Typically, the most advanced commercial engines allow to implement a UDF in any programming language and call it from the engine as an external program (i.e., as a pure black-box). For this reason, optimizing the execution of DI processes with BBUDFs is more than challenging. To be able to apply the aforementioned optimization techniques, one must know performance characteristics of BBUDFs and (if possible) their semantics.

4. Research hypothesis

The research hypothesis stated in this panel talk threefold.

First, we expect that the push-down technique applied to non-relational DSs will allow to increase performance of DI processes (i.e., reduce their execution time). Based on the developed execution cost models and implementation skeletons, it will be possible to push-down typical DI tasks into non-relational DSs. The question, however, is how push-down could benefit from machine learning (ML) techniques in the course of: (1) deciding whether a given task should be pushed down into a non-relational DS and (2) providing an efficient implementation of the task in the DS.

Second, we expect that it will be possible to build performance models of basic and complex BBUDFs (like those listed in [11]) by applying ML techniques. The models of BBUDFs will be assigned to performance classes provided by prediction models built from analyzing the performance models of known UDFs. For BBUDFs, their performance characteristics will be collected and they will be classified into one of the already known performance classes, thus allowing us to reason at least about an expected BBUDF performance. Our initial work [39] shows that the proposed approach is feasible on basic BBDUFs.

Third, we expect that it will be possible to build semantic models of basic BBUDFs by means of machine learning techniques - possibly by applying deep neural networks. Here, open question are: whether ML techniques could be used to build the models; what kind of techniques would be suitable; what input data would be required?

The discussion on the aforementioned challenges should be extended towards a broader scope of DI: (1) whether ML techniques can revolutionize the development and deployment methods of efficient DI pipelines, (2) how to build an end-to-end DI pipeline with the support of ML, (3) how to assure and verify the quality of data produced by such pipelines, (4) how to leverage the ML techniques for building complex DI architectures with appropriately designed software and hardware, and (5) how to mitigate bias in the ML techniques used to build DI pipelines. Furthermore, another question is whether ML could help solving the still unsolved challenge of the ETL evolution, e.g., [40, 41].

References

[1] A. Bouguettaya, B. Benatallah, A. Elmargamid, Interconnecting Heterogeneous Information Systems, Kluwer Academic Publishers, ISBN 0792382161, 1998.

[2] P. Brezany, A. M. Tjoa, H. Wanek, A. Wöhrer, Mediators in the architecture of grid information systems, in: Int. Conf. Parallel Processing and Applied Mathematics (PPAM), volume 3019 of *LNCS*, Springer, 2003, pp. 788–795.

[3] S. A. Errami, H. Hajji, K. A. E. Kadi, H. Badir, Spatial big data architecture: From data warehouses and data lakes to the lakehouse, Journal of Parallel and Distributed Computing 176 (2023) 70–79.

[4] A. A. Munshi, Y. A. I. Mohamed, Data lake lambda architecture for smart grids big data analytics, IEEE Access 6 (2018) 40463–40471.

[5] R. Hai, C. Koutras, C. Quix, M. Jarke, Data lakes: A survey of functions and systems, 2023. arXiv:2106.09592.

[6] A. A. Harby, F. H. Zulkernine, From data warehouse to lakehouse: A comparative review, in: IEEE Big Data, 2022, pp. 389–395.

[7] R. Tan, R. Chirkova, V. Gadepally, T. G. Mattson, Enabling query processing across heterogeneous data models: A survey, in: IEEE Big Data, 2017, pp. 3211–3220.

[8] Z. Dehghani, Data Mesh: Delivering Data-Driven Value at Scale, O'Reilly, ISBN 1492092398, 2022.

[9] T. Furche, G. Gottlob, L. Libkin, G. Orsi, N. W. Paton, Data wrangling for big data: Challenges and opportunities, in: Int. Conf. on Extending Database Technology (EDBT), 2016, pp. 473–478.

[10] A. Simitsis, S. Skiadopoulos, P. Vassiliadis, The history, present, and future of ETL technology (invited), in: Int. Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP) @EDBT/ICDT, volume 3369 of *CEUR Workshop Proceedings*, 2023, pp. 3–12.

[11] IBM, Product documentation: Infosphere information server 11.3, https://www.ibm.com/docs/en/iis/11.3?topic=jobs-processing-data, 2023.

[12] S. M. F. Ali, R. Wrembel, From conceptual design to performance optimization of ETL workflows: current state of research and open problems, The VLDB Journal 26 (2017) 777–801.

[13] Gartner, Magic quadrant for data integration tools, 2022.

[14] M. Birgersson, G. Hansson, U. Franke, Data integration using machine learning, in: IEEE Int. Enterprise Distributed Object Computing Workshop (EDOC), 2016, pp. 1–10.

[15] L. Dong, T. Rekatsinas, Data integration and machine learning: a natural synergy, Proc. VLDB Endowment 11 (2018) 2094–2097.

[16] I. F. Ilyas, T. Rekatsinas, Machine learning and data cleaning: Which serves the other?, ACM Journal of Data and Information Quality 14 (2022) 13:1–13:11.

[17] N. Barlaug, J. A. Gulla, Neural networks for entity matching: a survey, ACM Transactions on Knowledge Discovery from Data 15 (2021) 52:1–52:37.

[18] A. Zeakis, G. Papadakis, D. Skoutas, M. Koubarakis, Pre-trained embeddings for entity resolution: An experimental analysis, Proc. VLDB Endowment 16 (2023) 2225–2238.

[19] D. V. Aken, A. Pavlo, G. J. Gordon, B. Zhang, Automatic database management system tuning through large-scale machine learning, in: Int. Conf. on Management of Data (SIGMOD), 2017, pp. 1009–1024.

[20] M. Golfarelli, S. Graziani, S. Rizzi, An active learning approach to build adaptive cost models for web services, Data & Knowledge Engineering 119 (2019) 89–104.

[21] Á. B. Hernández, M. S. Pérez, S. Gupta, V. Muntés-Mulero, Using machine learning to optimize parallelism in big data applications, Future Generation Computer Systems 86 (2018) 1076–1092.

[22] S. Pumma, W. Feng, P. Phunchongharn, S. Chapeland, T. Achalakul, A runtime estimation framework for ALICE, Future Generation Computer Systems 72 (2017) 65–77.

[23] R. Sellami, B. Defude, Complex queries optimization and evaluation over relational and nosql data stores in cloud environments, IEEE Transactions on Big Data 4 (2018) 217–230.

[24] J. Taheri, A. Y. Zomaya, A. Kassler, vmbbprofiler: a black-box profiling approach to quantify sensitivity of virtual machines to shared cloud resources, Computing 99 (2017) 1149–1177.

[25] C. Witt, M. Bux, W. Gusew, U. Leser, Predictive performance modeling for distributed batch processing using black box monitoring and machine learning, Information Systems 82 (2019) 33–52.

[26] A. Pavlo, M. Butrovich, L. Ma, P. Menon, W. S. Lim, D. V. Aken, W. Zhang, Make your database system dream of electric sheep: Towards self-driving operation, Proc. VLDB Endowment 14 (2021) 3211–3221.

[27] T. Kraska, T. Li, S. Madden, M. Markakis, A. Ngom, Z. Wu, G. X. Yu, Check out the big brain on BRAD: simplifying cloud data processing with learned automated data meshes, Proc. VLDB Endowment 16 (2023) 3293–3301.

[28] S. M. F. Ali, J. Mey, M. Thiele, Parallelizing user–defined functions in the etl workflow using orchestration style sheets, Int. Journal of Applied Mathematics and Computer Science 29 (2019) 69–79.

[29] A. Karagiannis, P. Vassiliadis, A. Simitsis, Scheduling strategies for efficient etl execution, Information Systems 38 (2013) 927–945.

[30] X. Liu, N. Iftikhar, An ETL optimization framework using partitioning and parallelization, in: ACM Symposium on Applied Commputing (SAC), 2015, pp. 1015–1022.

[31] S. M. F. Ali, R. Wrembel, Framework to optimize data processing pipelines using performance metrics, in: Int. Conf. on Big Data Analytics and Knowledge Discovery (DAWAK), LNCS 12393, 2020, pp. 131–140.

[32] A. Simitsis, P. Vassiliadis, T. K. Sellis, State-space optimization of ETL workflows, IEEE Transactions on Knowledge and Data Engineering 17 (2005) 1404–1419.

[33] C. Yan, Y. Lin, Y. He, Predicate pushdown for data science pipelines, Int. Conf. on Management of Data (SIGMOD) 1 (2023).

[34] M. Bodziony, R. Morawski, R. Wrembel, Evaluating push-down on nosql data sources: experiments and analysis paper, in: Int. Workshop on Big Data in Emergent Distributed Environments(BiDEDE) @ ACM SIGMOD/PODS Conference, ACM, 2022, pp. 4:1–4:6.

[35] M. Bodziony, S. Roszyk, R. Wrembel, On evaluating performance of balanced optimization of ETL pro-

cesses for streaming data sources, in: Int. Workshop on Design, Optimization, Languages and Analytical Processing of Big Data (DOLAP) @EDBT/ICDT, volume 2572 of *CEUR Workshop Proceedings*, 2020, pp. 74–78.

[36] C. Forresi, M. Francia, E. Gallinucci, M. Golfarelli, Cost-based optimization of multistore query plans, Information Systems Frontiers 25 (2023) 1925–1951.

[37] A. Crotty, A. Galakatos, K. Dursun, T. Kraska, C. Binnig, U. Cetintemel, S. Zdonik, An architecture for compiling udf-centric workflows, VLDB Endowment 8 (2015) 1466–1477.

[38] Q. Chen, R. Wu, M. Hsu, B. Zhang, Extend core UDF framework for gpu-enabled analytical query evaluation, in: Int. Database Engineering and Applications Symposium (IDEAS), 2011, pp. 143–151.

[39] A. Lehnhardt, B. Ciesielski, Designing and implementing a method for assessing similarities between time series on computer resources consumed by data processing tasks, Master thesis, Poznan University of Technology, 2022.

[40] D. Butkevicius, P. D. Freiberger, F. M. Halberg, MAIME: A maintenance manager for ETL processes, in: Workshops of the EDBT/ICDT Joint Conference, volume 1810 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2017.

[41] G. Papastefanatos, P. Vassiliadis, A. Simitsis, Y. Vassiliou, Policy-regulated management of ETL evolution, Journal on Data Semantics 13 (2009) 147–177.