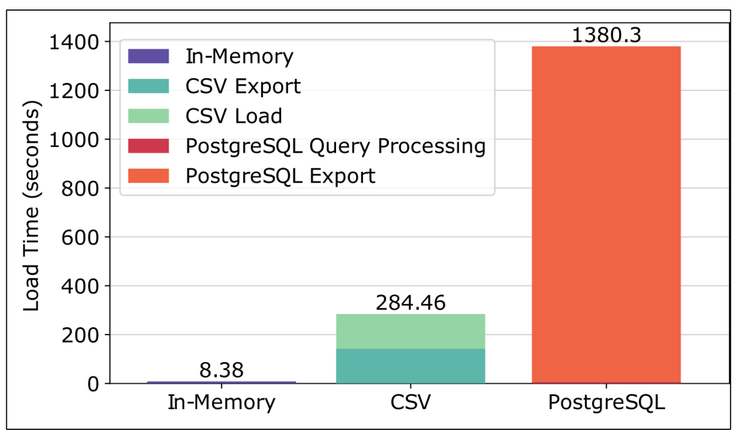
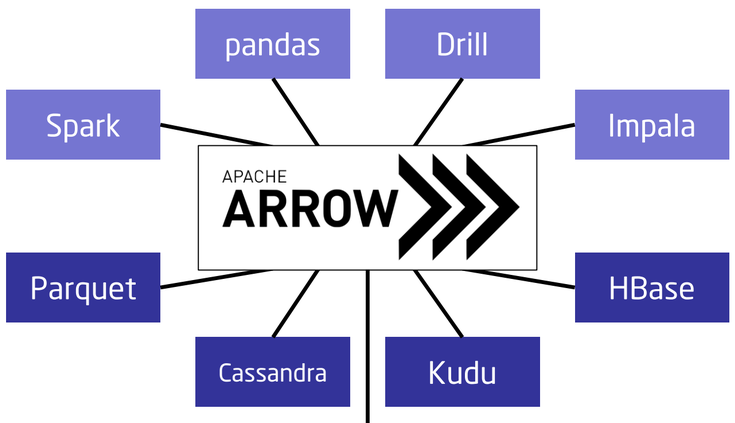
Data Without Borders: RDBMSs and Universal Columnar Memory Layouts

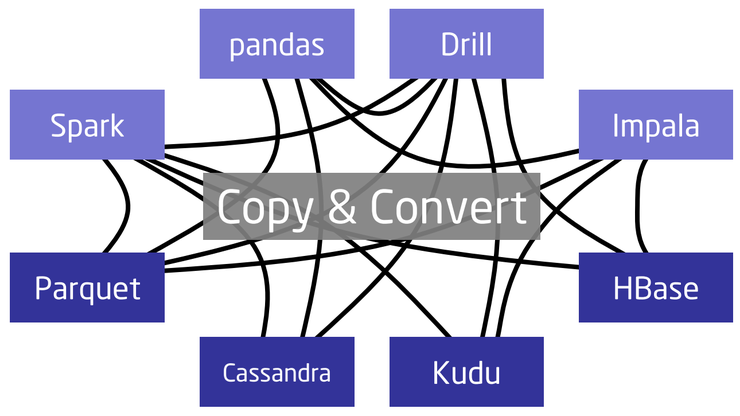


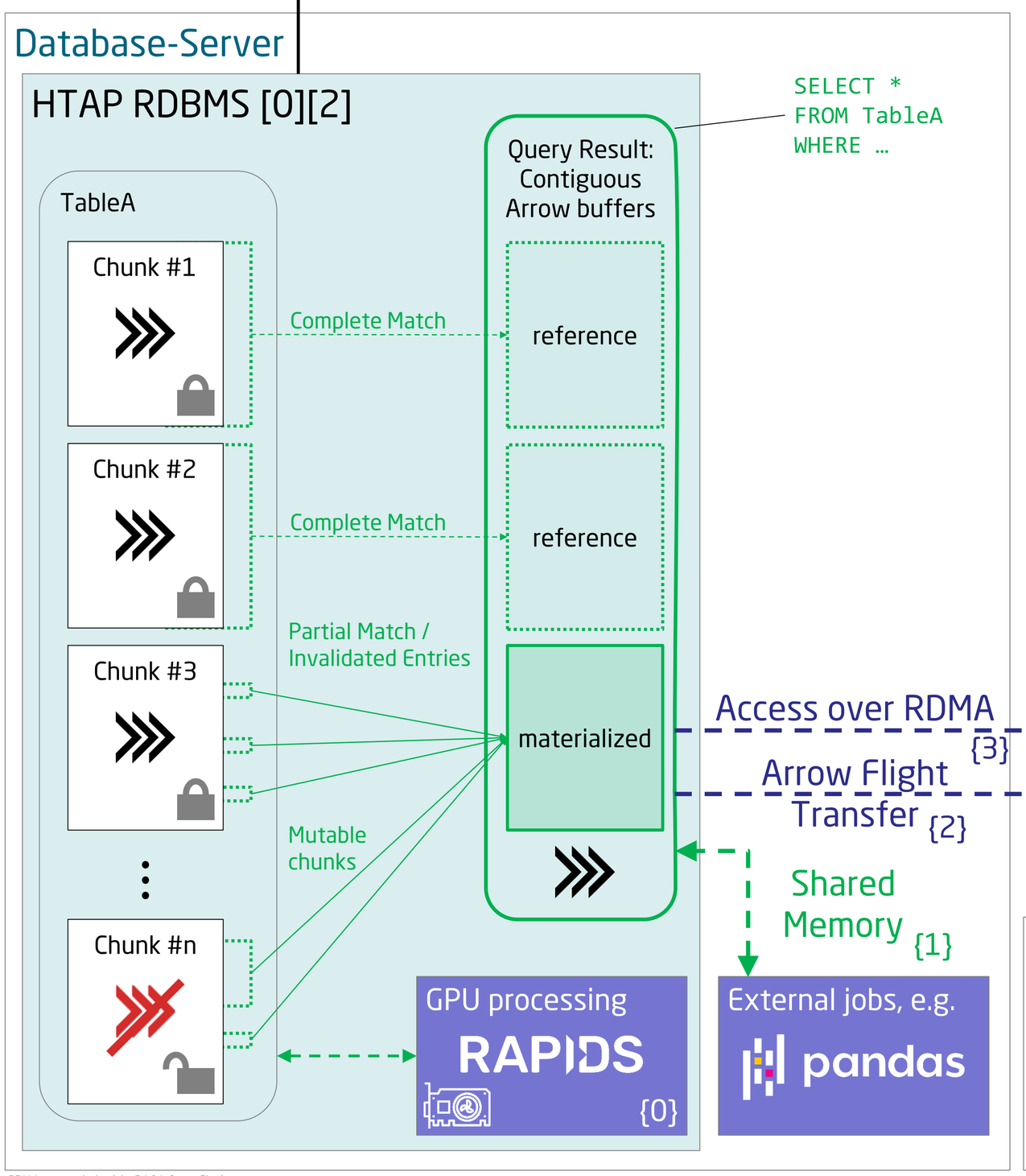
Varying transfer times of a 8GB TPC-H table into pandas. Borrowed from Li [0].

Databases and Data Engineering: Many data pipelines incorporate relational databases, either as an easily queryable multi-user analytical data store or as a single source of truth for transactional data. However, mainstay tools for data preprocessing, domain-specific transformations or advanced analytical calculations (e.g. hardware-accelerated machine learning algorithms) revolve heavily around external ecosystems such as Python. Transferring data between these systems brings costly serialization/deserialization from one data representation to the other [0][1]. This hinders quick iteration in data exploration phases and wastes power on unnecessary computation in automated pipelines.



Apache Arrow provides a standardized, language-agnostic memory layout for columnar data, designed for analytic processing on modern hardware. Applications adhering to the standard can zero-copy exchange data, eliminating serialization overhead. Even though it was designed for analytic read-only processing, recent work by Li et al. has shown the feasibility of transaction engines backed by Arrow memory [0].





GPU Icon made by itim2101 from Flaticon

A Universal DBMS? We propose a wide-spanning examination of Arrow-backed HTAP systems which allow zero-copy and/or serialization-less data exchange with database clients. This work might start with adapting an open-source database (e.g. [2]) to utilize Arrow and investigating performance impacts on benchmarks such as TPC-H. Opening up application-specific memory formats would allow the direct or indirect integration of various open-source analytics libraries: Potentially, chunk-wise query operators could be sped up by GPU-accelerated processing provided by cuDF {0}. Also, user-defined jobs based on Arrow-compliant libraries could be executed directly on the results of analytical queries. This might be achieved through direct execution on the database host with shared memory between job process and DBMS {1}. For this scenario, practical concerns such as security, resource allocation, access control and data locking should be investigated. Furthermore, more efficient data exchange to external clients can be achieved, either through no-serialization transfer {2} or RDMA {3} in case corresponding hardware is available. On the client's side, the transferred data is directly useable by libraries utilizing the standardized format. Overall, these optimizations could provide substantial increases in efficiency for various types of data pipelines.

Philipp Bode
Master Student IT-Systems-Engineering

Hasso Plattner Institute, Potsdam, Germany

E-Mail: philipp.bode@student.hpi.de

[0] Li, T. (2019). Supporting Hybrid Workloads for In-Memory Database Management Systems via a Universal Columnar Storage Format. *Master Thesis, CMU Database Group, Prof. Andrew Pavlo*

[1] Raasveldt, M., & Mühleisen, H. (2017). Don't hold my data hostage-A case for client protocol redesign. *Proceedings of the VLDB Endowment*, *10*(10), 1022–1033.

[2] Dreseler, M., Kossmann, J., Boissier, M., Klauck, S., Uflacker, M., & Plattner, H. (2019). Hyrise re-engineered: An extensible database system for research in relational in-memory data management. *Advances in Database Technology - EDBT*, *2019-March*, 313–324.

Hasso Plattner Institut