

Hasso Plattner Institute

Chair for Data Engineering Systems



Proposal Master Thesis

Using Column Stores for Stream Processing

Björn Daase

Time frame: August 2022 - January 2023

Supervisor

Prof. Dr. Tilmann Rabl

Advisor

Lawrence Benson, M. Sc.

1 Motivation

Current Stream Processing Engines (SPEs) can process terabytes of incoming data per second [2]. Because widely used SPEs such as Apache Flink and Spark Streaming do not fully utilize the underlying hardware and are resource inefficient [14, 15], the implementation of recent SPEs shifted towards system languages such as C and C++. This increased their throughput by up to two orders of magnitude compared to state-of-the-art SPEs [14].

However, along with these performance gains, the bottlenecks of SPEs have also shifted. In particular, it has become apparent that memory and not CPU performance is the new bottleneck for many queries [4].

The shift to system languages also brings SPEs much closer to classical database systems, which are usually also implemented in system languages. One trend observed in classical database systems in recent years is the introduction of more and more column-oriented database systems, like MonetDB [5, 7, 9], C-Store [13] or SAP HANA [12]. They work around the same memory bottleneck we now observe for SPEs [6] by accessing memory more efficiently [1] which makes them ultimately faster than row-based database systems.

However, this column-based store seems inherently incompatible with true streaming, which assumes processing one tuple at a time. However, this understanding no longer corresponds to the truth, since tuples arrive at least in mini-batches corresponding to the size of network packets and SPEs process them as is. Accordingly, it is interesting to investigate how a column-oriented SPE performs against a row-based one (or even hybrid forms). With MonetDB/Datacell [11] and Trill [8], two SPEs have been designed in recent years that operate on a column-store. Of particular interest is how the characteristics of the queries and tuples influence the performance.

2 Goal of Thesis

In order to understand the impact of a column-based compared to a row-based layout for an SPEs, we analyze their raw query performance along several microbenchmarking dimensions. Therefore, we store the data as an Array of Structs (AoS) when simulating a row-store and as an Struct of Arrays (SoA) in the case of a column store and analyze their performance across an exhausting set of queries. Independently of this, we will look at the transformation of incoming tuples into a columnar-based store as well as their materialization back into a row-based format at the end of the query. This also raises the question of whether hybrid forms of data storage are worthwhile. If, for example, some fields are not used during the actual processing (i.e. no filters, joins or aggregations take place on them), but they have to be finally issued with the tuples as payload, this cold data could be kept in a row-store while the hot data is processed in a column-store.

An additional goal will be to also understand how these different approaches perform on different hardware. As shown by Kersten et al. [10], vectorizing plays a central role in efficient query processing. However, with different vectorizing extensions on different platforms as well as different kinds of memory on different machines [4], especially the aforementioned High Bandwidth Memory (HBM), the performance of row-store-based and column-store-based streaming could be influenced significantly. For reference, the HBM ARM systems can reach up to 3x sequential memory bandwidth compared to state-of-the-art Intel, AMD, and Power systems which peak around 200 GB/s [4].

As a result, we can hopefully create a cost model that decides at query compilation time whether it will process data more efficiently row-based, column-based, or hybrid. Finally, we test the suitability of our cost model and non-row-store streaming in the Darwin stream processing engine [3].

Given that, we plan the following contributions:

1. We understand the impact of different data layouts (row-based, column-based, or hybrid) on the query performance of SPEs.
2. We create a cost model that allows to pick the best layout at query compilation time and validate that in the Darwin stream processing engine.

3 Approach

For the microbenchmarks, we first focus on **(a)** the used query primitives (projections, filters, joins, aggregations), **(b)** its selection as well as **(c)** its selectivity. We follow along with **(d)**, the size of the schema in combination with **(e)**, the data types used in the schema (mainly in-line types, i.e. *int* vs. pointer indirect types, i.e. *std::string*). Finally, we analyze how **(f)** the mini-batch size, different **(g)** window types (mainly tumbling and sliding windows) and if time permits **(h)** multithreading influences the result.

3.1 Query Primitives

Relational algebra boils down to the following operators: *Projections*, *Filters*, *Joins*, *Aggregations*, and *Selections*. In this section, we will analyze all of the above except selections, since they are somehow special when comparing row-based and column-based streaming since each operation represents a clean filter in one dimension.

3.2 Selection

Excluded from the classical relational algebra primitives we consider the *Selection* as well as the *Selectivity*. A high selectivity should be favorable for the row-store

since this would allow aligned erasing of data out of an AoS. On the other hand, working only on a few columns allows dropping entire arrays from an AoS.

3.3 Selectivity

3.4 Schema Size

3.5 Schema Data Types

3.6 Mini-batch Size

3.7 Window Types

3.8 Multithreading

4 Related Work

Description of related work in literature or other projects (theses, master projects, etc.).

4.1 Stream Processing Engines

4.2 Column Stores

4.3 Memory Performance of High-Performance Computers

5 Project Plan

Please find a timetable with major milestones here.

Time	Writing/Research	Prototype
August	Write Background	
Apr - May	<ul style="list-style-type: none"> – Windowing – Stream slicing – Aggregate sharing 	<ul style="list-style-type: none"> – Get Scotty up and running – First concept of distributed windows – Distributive aggregate functions (e.g. <code>count</code>)
May - Sep	<ul style="list-style-type: none"> – Distributed slices – Distributed sharing – Processing semantics 	<ul style="list-style-type: none"> – Different window types – Aggregate sharing – Algebraic and holistic functions (e.g. average, median)
Aug - Oct	– Evaluation	– Implement alternatives

Table 1: Planned Time Table

References

- [1] Daniel J. Abadi, Samuel Madden, and Nabil Hachem. Column-stores vs. row-stores: how different are they really? In Jason Tsong-Li Wang, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 967–980. ACM, 2008. doi: 10.1145/1376616.1376712. URL <https://doi.org/10.1145/1376616.1376712>.
- [2] Alibaba. Four billion records per second! www.alibabacloud.com/blog/four-billion-records-per-second-stream-batch-integration-implementation-of-alibaba-cloud-realtime-compute-for-apache-flink-during-double-11-596962, 2020.
- [3] Lawrence Benson and Tilmann Rabl. Darwin: Scale-in stream processing. In *12th Conference on Innovative Data Systems Research, CIDR 2022, Chaminade, CA, USA, January 9-12, 2022*. www.cidrdb.org, 2022. URL <https://www.cidrdb.org/cidr2022/papers/p34-benson.pdf>.
- [4] Lars Jonas Bollmeier, Björn Daase, and Richard Ebeling. Processor-specific stream processing query compilation. Technical report, Hasso Plattner Institute, University of Potsdam, 2021.
- [5] Peter A. Boncz and Martin L. Kersten. MIL primitives for querying a fragmented world. *VLDB J.*, 8(2):101–119, 1999. doi: 10.1007/s007780050076. URL <https://doi.org/10.1007/s007780050076>.
- [6] Peter A. Boncz, Stefan Manegold, and Martin L. Kersten. Database architecture optimized for the new bottleneck: Memory access. In Malcolm P. Atkinson, Maria E. Orlowska, Patrick Valduriez, Stanley B. Zdonik, and Michael L. Brodie, editors, *VLDB’99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 54–65. Morgan Kaufmann, 1999. URL <http://www.vldb.org/conf/1999/P5.pdf>.
- [7] Peter A. Boncz, Marcin Zukowski, and Niels Nes. Monetdb/x100: Hyper-pipelining query execution. In *Second Biennial Conference on Innovative Data Systems Research, CIDR 2005, Asilomar, CA, USA, January 4-7, 2005, Online Proceedings*, pages 225–237. www.cidrdb.org, 2005. URL <http://cidrdb.org/cidr2005/papers/P19.pdf>.
- [8] Badrish Chandramouli, Jonathan Goldstein, Mike Barnett, Robert DeLine, John C. Platt, James F. Terwilliger, and John Wernsing. Trill: A high-performance incremental query processor for diverse analytics. *Proc. VLDB Endow.*, 8(4):401–412, 2014. doi: 10.14778/2735496.2735503. URL <http://www.vldb.org/pvldb/vol8/p401-cchandramouli.pdf>.
- [9] Stratos Idreos, Fabian Groffen, Niels Nes, Stefan Manegold, K. Sjoerd Mullender, and Martin L. Kersten. Monetdb: Two decades of research in column-oriented database architectures. *IEEE Data Eng. Bull.*, 35(1):40–45, 2012. URL <http://sites.computer.org/debull/A12mar/monetdb.pdf>.
- [10] Timo Kersten, Viktor Leis, Alfons Kemper, Thomas Neumann, Andrew Pavlo, and Peter A. Boncz. Everything you always wanted to know about compiled and vectorized queries but were afraid to ask. *Proc. VLDB Endow.*, 11(13):2209–2222, 2018. doi: 10.14778/3275366.3275370. URL <http://www.vldb.org/pvldb/vol11/p2209>

-kersten.pdf.

- [11] Erietta Liarou, Stratos Idreos, Stefan Manegold, and Martin L. Kersten. Monetdb/datacell: Online analytics in a streaming column-store. *Proc. VLDB Endow.*, 5(12):1910–1913, 2012. doi: 10.14778/2367502.2367535. URL http://vldb.org/pvldb/vol5/p1910_eriettaliarou_vldb2012.pdf.
- [12] Vishal Sikka, Franz Färber, Wolfgang Lehner, Sang Kyun Cha, Thomas Peh, and Christof Bornhövd. Efficient transaction processing in SAP HANA database: the end of a column store myth. In K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 731–742. ACM, 2012. doi: 10.1145/2213836.2213946. URL <https://doi.org/10.1145/2213836.2213946>.
- [13] Michael Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Samuel Madden, Elizabeth J. O’Neil, Patrick E. O’Neil, Alex Rasin, Nga Tran, and Stanley B. Zdonik. C-store: A column-oriented DBMS. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 553–564. ACM, 2005. URL <http://www.vldb.org/archives/website/2005/program/paper/thu/p553-stonebraker.pdf>.
- [14] Steffen Zeuch, Sebastian Breß, Tilmann Rabl, Bonaventura Del Monte, Jeyhun Kariyov, Clemens Lutz, Manuel Renz, Jonas Traub, and Volker Markl. Analyzing efficient stream processing on modern hardware. *Proc. VLDB Endow.*, 12(5):516–530, 2019. doi: 10.14778/3303753.3303758. URL <http://www.vldb.org/pvldb/vol12/p516-zeuch.pdf>.
- [15] Shuhao Zhang, Bingsheng He, Daniel Dahlmeier, Amelie Chi Zhou, and Thomas Heinze. Revisiting the design of data stream processing systems on multi-core processors. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 659–670. IEEE Computer Society, 2017. doi: 10.1109/ICDE.2017.119. URL <https://doi.org/10.1109/ICDE.2017.119>.