Build your own OctopusDB: Blinktopus Edition

Pavlo Shevchenko, Ali Hashaam, Guzel Mussilova, Ali Memon
Otto-von-Guericke-University, Magdeburg
firstname.lastname@st.ovgu.de

***Abstract*—What is this paper about?**

I. Introduction

Over the last decades we are witnessing that modern enterprises need to pick only specialized DBMSs(e.g. OLAP, OLTP, streaming systems and etc.) each tailored to their specific use-case. Consequently, it leads to additional costs in terms of licensing, maintenance, integration and man-hours for DBAs. Although, it is affordable for some companies, to adapt these integrated solutions to constantly changing workloads and requirements. However, it may still be a challenging and non-trivial task to achieve. Thus, in order to cope with these problems an implementation of new all-purpose system will be a perfect solution.

Nowadays there exists a great variety of systems that claim to solve the aforementioned problems. The traditional DBMSs(e.g., Microsoft SQL Server, Oracle, ...) have already included the support of both analytical (OLAP, which is characterized by long-running queries over all the values of few columns) and transactional (OLTP, characterized by short-lived transactions that affect all attributes of few rows) workloads. At the same time, they are not that efficient. Thus, past years the *one size doesn't fit all* rule has lead the various vendors to build the specialized solutions that exploiting the progress in modern hardware[6]. Among these in-memory optimized solutions are the column-stores(e.g., C-Store, MonetDB[8], ...) and the row-stores(e.g., Hekaton[7], H-Store, ...) that particularly designed for analytical and transactional processing, respectively. Later in order to extend the functionality and to support Hybrid Transactional/Analytical Processing (HTAP) as well, the various systems tailored to one type of processing started supporting the other type. For instance, SAP HANA[9] has the engines that are optimized for OLAP workloads, at the same time it also support ACID transactions. As for HyPer[10], which primarily was using row-wise processing to support both types of workloads, now it also presented the opportunity to use a columnar format to run analytical requests in a more efficient manner. The following examples such as Peloton[11], H2O, OctopusDB[1] and SnappyData also belong to HTAP systems. One of the solutions that most radically departs from existing architectures was proposed by Jens Dittrich and Alekh Jindal - a new type of database system, coined OctopusDB[1]. By mimicking several types of systems, OctopusDB shows a considerably better performance. Meanwhile, it became evident that a system performance can be further enhanced by applying techniques from completely different perspectives. As long as it is plausible for a system to retrieve results approximately rather than exactly, Approximate Query Processing(AQP) could be a reasonably good fit to further improve the performance of query processing (especially, OLAP queries) and it could also enhance HTAP by answering the OLAP queries over new data that ever has not been included yet. Several successful examples(e.g. BlinkDB, SnappyData on Facebook's Presto) have already proved that there are benefits to be gained by integrating approximation features into existing DBMSs[5]. We believe that combining OctopusDB and AQP techniques could be feasibly good solution for improving HTAP *freshness* and the performance of our system. Thus, our goal is to provide a user a tool called Blinktopus, that will allow to build his own prototype of OctopusDB with embedded AQP techniques.

In Section 2 we start with the brief discussion of what is the idea behind OctopusDB and AQP. Moreover, in Section 2 we introduce the concepts of AQP main data synopses and explain why exactly one type of synopsis was chosen for our system. We present our contributions as follows:

- In Section 3 we propose a novel concept of a new system called Blinktopus.
- In Section 4 we discuss the experimental part of our project. Here we provide an evaluation on the benefits of Blinktopus's functionality based on the results of the tests performed on the Storage Views(SVs) with the different physical layouts, namely LogSV, ColSV and RowSV.

In Section 5 we present the related work. We conclude this paper in Section 6 and propose future directions for our research in Section 7.

II. Fundamentals

A. OctopusDB

Main Idea. Architecture briefly.

B. Approximate Query Processing

Main Idea. Synopses for Massive Data: Histograms. Sketches. More on histograms. Probably something on HLL, if it's eventually included in Blinktopus.

III. Blinktopus

IV. Experimental Part

Evaluation of the test results:

1. Comparison of runtimes of different types of SVs. 2. Comparison of the diff SVs by means of percentage representation.

V. Related Work

Some related work has to be mentioned here.

VI. Conclusions

Sum up what happened

VII. Future Work

Acknowledgement

Thanks to Gabriel(DBSE). Also we would like to thank Jindal, Dittrich and Mozafari for their awesome ideas that really inspired this project.

References

[1] Dittrich, Jens, and Alekh Jindal. "Towards a One Size Fits All Database Architecture." CIDR. 2011.

[2] Jindal, Alekh. "OctopusDB: flexible and scalable storage management for arbitrary database engines." (2012).

[3] Jindal, Alekh. "The mimicking octopus: Towards a one-size-fits-all database architecture." VLDB PhD Workshop. 2010.

[4] Mozafari, Barzan. "Approximate query engines: Commercial challenges and research opportunities." SIGMOD, 2017.

[5] Mozafari, Barzan, and Ning Niu. "A Handbook for Building an Approximate Query Engine." IEEE Data Eng. Bull. 38, no. 3 (2015): 3-29.

[6] M. Stonebraker and U. Cetintemel. "One Size Fits All": An Idea Whose Time Has Come and Gone. In ICDE, pages 211, 2005.

[7] C. Diaconu, C. Freedman, E. Ismert, P.-A. Larson, P. Mittal, R. Stonecipher, N. Verma, and M. Zwilling. Hekaton: SQL Servers memory-optimized OLTP engine. In SIGMOD, pages 12431254, 2013.

[8] P. Boncz, M. Zukowski, and N. Nes. MonetDB/X100: Hyper-Pipelining Query Execution. In CIDR, 2005.

[9] F. Frber, N. May, W. Lehner, P. Groe, I. Mller, H. Rauhe, and J. Dees. The SAP HANA Database An Architecture Overview. IEEE DEBull 35(1):2833, 2012.

[10] A. Kemper and T. Neumann. HyPer A Hybrid OLTP'&'OLAP Main Memory Database System Based on Virtual Memory Snapshots. In ICDE, pages 195206, 2011.

[11] A. Pavlo, J. Arulraj, L. Ma, P. Menon, T. C. Mowry, M. Perron, A. Tomasic, D. V. Aken, Z. Wang, and T. Zhang. Self-Driving Database Management Systems. In CIDR, 2017.