Operational Analytics Data Management Systems

Alexander Böhm
SAP SE

Jens Dittrich
Saarland University

Niloy Mukherjee
Oracle

Ippokratis Pandis
Amazon Web Services

Rajkumar Sen
Oracle

1. INTRODUCTION

Prior to mid-2000s, the space of data analytics was mainly confined within the area of *decision support systems.* It was a long era of isolated enterprise data warehouses curating information from live data sources and of business intelligence software used to query such information. Most data sets were small enough in volume and static enough in velocity to be segregated in warehouses for analysis. Data analysis was not ad-hoc; it required pre-requisite knowledge of underlying data access patterns for the creation of specialized access methods (e.g. covering indexes, materialized views) in order to efficiently execute a set of few focused queries.

The last decade witnessed a rapid overhaul in the area of business analytics. With the advent of ubiquitous data sources resulting in unprecedented explosion in ingestion volumes, analytic database systems had to evolve on multiple fronts. They were now required to provide high performance query processing over large volumes of data, handle ad-hoc queries, scale with the growing data volumes, excel in performance on clusters of commodity hardware, and last but not the least, capture very specific real-time analytic insights in live mainstream production environments.

The decade long evolution of analytic databases has been paved with several technical milestones. Early to mid 2000s witnessed the emergence of MPP OLAP appliances (e.g. Teradata, Netezza, Exadata, Exasol) along with the resurgence of columnar data models (e.g. Actian Vector, Vertica) that were both capacity and compute-friendly. These appliances were multi-server systems with hundreds of computing cores and terabytes of storage. They came with integrated database management software that provided high performance query throughput on large volumes of data typically at rest. The same period also witnessed the dramatic rise of social and mobile applications that began generating volumes of unstructured raw data. Software frameworks such as Mapreduce and Hadoop paved the way for a new generation of analytic data management systems that batch-processed vast amounts of at-rest data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. Wider adoption of these technologies in the enterprise required adhering to SQL standards, which led SQL processing over Hadoop-like frameworks to gain significant traction in the last few years.

Although these technologies were able to provide query processing over large volumes of data, they operated mostly on data at rest. As enterprise businesses become more agile, they demand real time insights as data gets ingested and modified in live mainstream OLTP-like production environments. Let us take the scenario of machine data analytics for example. A typical cloud-scale enterprise data center generates several terabytes of metrics data per day from its applications, middleware, servers, virtual machines, and fiber ports. In order to proactively mitigate risks and attain quick insights to identify and resolve non-predictive events in real time, such environments require high performance ad-hoc query processing over multiple metrics in real-time over large volumes of data constantly being ingested from multiple sources. Social media based retail analytics serves as another relevant example. Such applications require analytic insights on immediate surges of interest on social media platforms to derive targeted product trends in real-time.

Business applications like these would not afford typical ETL-like lag of moving terabytes of data from OLTP-like sources to at-rest data warehouses. These emerging breed of applications have therefore necessitated the evolution of data management systems to focus on providing breakthrough analytics performance not only in traditional at-rest OLAP environments but also in mixed *online transactional analytics processing* (henceforth, *OLTAP*) environments where operational data gets continuously ingested and modified. Henceforth, we will refer to these systems as *operational analytics* systems.

Last but not the least, there has been resurgence in main-memory based analytic database systems in the last few years. Today's multi-core, multiprocessor servers provide fast communication between processor cores via main memory, taking full advantages of main memory bandwidths, and alleviating storage I/O bottlenecks incurred in query processing throughput. Main memory is therefore being conceived by DBMS architects more as a primary storage container and less as a cache optimizing disk based accesses.

In this tutorial, we will examine operational analytics systems along various dimensions. This includes, among others:

- Physical data storage (row-based, column-oriented, or hybrid) and different access paths

Proceedings of the VLDB Endowment, Vol. 9, No. 13
Copyright 2016 VLDB Endowment 2150-8097/16/09.

- Design choices for corresponding runtime systems (e.g. efficient query execution and transaction management)
- Advanced query processing topics like NUMA-awareness and shared scans
- Workload management for complex, mixed (OLTP and OLAP) workloads
- System scalability by scaling up (to large NUMA machines) and scaling out (to distributed deployments)

Operational analytics is an area that has been widely researched in streaming systems; both in academia and industry, e.g. [1, 21]. Traditionally, streaming systems have not been known to guarantee ACID properties for both transactional and analytic workloads (with exceptions, such as [25]). At the same time their analytics capabilities are typically fairly limited. In this tutorial, the focus is on systems that guarantee ACID properties and can serve a wide range of analytic reporting workloads. Therefore, we would not consider any streaming system.

2. TUTORIAL STRUCTURE

In this 3-hour tutorial, we will first discuss the general system characteristics, and examine different approaches to data storage, query processing and optimization, indexing, and updates. In the second half of the tutorial, we will examine a set of representative systems in detail, highlight their individual architecture and design characteristics, and discuss several of the key research problems they address.

3. PERSPECTIVE OF INDUSTRY

Given the limited time duration, we present a selected number of systems that comprise the state-of-the-art in OLTAP. We start the presentation with SAP HANA, which was one of the first systems that focused on this particular workload. We present systems by both major database vendors, such as SAP, IBM and Oracle, as well as from startups, like MemSQL and Cloudera.

SAP HANA [35] is an in-memory database management system designed for enabling flexible and ad-hoc real-time reporting in enterprise systems [30]. The system allows to run both transactional and analytical workloads on a single, dictionary-compressed, in-memory column store. Its key characteristics include the heavy use of SIMD operations for highly efficient scans [42], optimizations for large NUMA systems [31], and support for complex, mixed workloads [32].

IBM DB2 with BLU Acceleration is a high-performing order-preserving dictionary-compressed columnar engine within DB2 [34]. Similar to SAP HANA, BLU employs SIMD and NUMA optimizations. In addition to having a high performance bulk data load utility, DB2 BLU fully supports INSERT, DELETE, and UPDATE statements, as well as the multi-threaded continuous INGEST utility. DB2 BLU is a multi-versioned data store in which deletes are logical operations that retain the old version rows and updates create new versions. Multiversioning enables DB2 BLU to support standard SQL isolation levels with minimal row locking.

Oracle Database In-memory [22] is a dual-format in-memory database management system to address ad-hoc real-time analytics in mixed OLTAP, workloads as well as traditional OLAP ones. The system allows both row and column format to be maintained at the level of an Oracle table, table partition, or composite partition [27]. While OLTP data manipulation and OLTP style queries are driven through the row format, analytic workloads are driven through the column format. Both formats are simultaneously active and strict transactional consistency is guaranteed between them in real-time. The in-memory columnar format inherits several compute and capacity utilization benefits of columnar processing, such as SIMD based vector processing, in-memory storage indexes, predicate evaluation push down, etc. The architecture enables application transparent distribution of in-memory columnar format across NUMA nodes within a single server as well as across a cluster of RDBMS servers, allowing for in-memory capacity and query-processing scale out, NUMA-aware scale-up [23], and high availability of the in-memory columnar format.

MemSQL is a distributed database designed to handle OLTP, OLAP and real-time streaming workloads with subsecond processing times in a single scalable database. The database engine features a row store in DRAM and a column store on flash/disk in a single database instance that allows low latency execution while still allowing for data growth. The row store is based on a lock free skip list [26] implementation that ensures high throughput for OLTP applications. The column store [36] is designed to support real-time streaming workloads while still leveraging all the query execution benefits of a compressed column store engine.

Cloudera uses Impala open-source, fully-integrated MPP SQL query engine for high-performing complex analytics in data lakes [20]. Unlike other systems (often forks of Postgres), Impala is a brand-new engine that employs LLVM to generate code at runtime to speed up frequently executed code paths [41]. In order to serve OLTAP workloads the users have a new option: have Impala query data stored in Kudu [24]. Kudu is an open source storage engine only for structured data which supports low-latency random access together with efficient scans. Kudu distributes data using horizontal partitioning and replicates each partition using Raft consensus. Kudu offers a simple API for row-level inserts, updates, and deletes, while providing table scans at throughput similar to Parquet.

4. ACADEMIC PERSPECTIVE

So far we have concentrated on the industry perspective. What has happened in academia?

For analytical queries, database research detected at least in 1979 already that a columnar layout, be it that you call it transposed files [4] or decompositional storage [7], is a great foundation for an analytical database. The earliest academic and to this data most influential system that consequently picked up that idea as its storage model was MonetDB from CWI. MonetDB is an open source system and also influenced commercial systems like Actian Vector. After MonetDB went open source in the early 2000s, we witnessed an abundance of papers dealing with the different aspects of column stores, including compression [15], tuple reconstruction [2], and updates [14]. That discussion also influenced several industry projects in their attitude that "columns are bad for transactions" and "rows are great for transactions". The mindset led several architects in designing systems that keep data in both row and column-layout at the same time. This is turn can be regarded a reincarnation of another idea from academia: fractured mirrors [33]. This mindset was also fostered by Mike Stonebraker who coined the mantra "One size does not fit all!" [37]. Other people both from industry

[30] and academia [8] argued against this in favor of building a single system where one (or at least few) size(s) fits all.

Naturally, once you have data in multiple copies every system designer runs into consistency and update problems. The most important technique in this spirit are differential files and LSM-style merge trees [29, 16], e.g. if you consider your column-store to be a read-only database and collect updates in a writable row-store, eventually you have to merge the two substores. So in a way this could be seen as pushing techniques previously implemented in two separate systems (column store: OLAP system, row store OLTP system) into a single system.

Eventually, academia also explored so-called hybrid data layouts. Those layouts may be vertically partitioned (aka column-grouped) layouts [17], data morphing [11], horizontally partitioned layouts [3] (which heavily influenced Apache Parquet), or any other combination [13]. Those layouts were also heavily explored in the context of Hadoop MapReduce, e.g. [18].

For transactions, there has also been a lot of interesting work in recent years, partially completely overturning how transactions are handled, e.g. H-store [38] proposed to prepartition the database into conflict-free partitions and run transactions in serial mode on each partition.

In terms of academic systems, the most interesting one in terms of mixed workloads is Hyper [19] from TU Munich. In their seminal work they built a system that is able to run an OLTP and and OLAP workload on the same system at the same time. The core idea is to create snapshots with the help of virtual memory. In [19] they showed how this can lead to breakthrough performance. They also proposed a suitable benchmark which combines TPC-C and TPC-H into a single CH-Benchmark [6]. In later work, among many other things, they also explored just-in-time LLVM code generation [28] which until then had been neglected in several systems [40] and now is considered state-of-the-art in query compilation.

Academia also developed a couple of 'fancy' database architectures that approached query processing from new angles. Prominent examples include the idea of a circular scan where incoming queries are attached to that scan [12]. This idea later evolved into a clock scan [39]. Other researchers explored the idea of having a more complex query pipeline where all data would be routed to depending on the queries currently subscribing to that plan [5, 9, 10].

5. TARGET AUDIENCE

Our target audience are database management system experts with an academic background as well as interested practitioners from the industry. Apart from fundamental knowledge about the architecture, design, and implementation aspects of database management systems and typical workloads that the typical VLDB audience generally has, there is no specific prior knowledge necessary.

6. TUTORIAL DIFFERENTIATION

This tutorial on Operational Analytics Data Management Systems has not been presented in any other venue, neither as is nor in any other, related structure. We believe that also the proposed setup is unique in a way that it is jointly given by a group of system architects from various corporations while at the same time also incorporating an academic perspective.

7. BIOGRAPHIES OF PRESENTERS

Alexander Böhm is a database architect working on SAP's HANA in-memory database management system. His focus is on performance optimization and holistic improvements of enterprise architectures, in particular application server/DBMS co-design. Prior to joining SAP, he received his PhD from the University of Mannheim, Germany, where he worked on the development of efficient and scalable applications using declarative message processing.

Jens Dittrich is a full professor of Computer Science in the area of Databases, Data Management, and Big Data at Saarland University, Germany. Previous affiliations include U Marburg, SAP AG, and ETH Zurich. He received an Outrageous Ideas and Vision Paper Award at CIDR 2011, a BMBF VIP Grant in 2011, a best paper award at VLDB 2014, two CS teaching awards in 2011 and 2013, as well as several presentation awards including a qualification for the interdisciplinary German science slam finals in 2012 as well as three presentation awards at CIDR (2011, 2013, and 2015). He likes producing educational database videos (`http://youtube.com/jensdit`) and flipped textbooks (`http://amzn.to/1Ts3rwx`).

Niloy Mukherjee is a consulting member of technical staff at Oracle RDBMS Data and In-memory Technologies working on distributed, absolute available, ACID compliant RDBMS architectures. He is one of the primary architects of Oracle Database In-memory Option, a fully consistent dual-format distributed in-memory RDBMS aimed to provide real-time analytics at scale on traditional OLAP as well as on mixed OLTAP workloads. He has published and presented his work at several VLDB, SIGMOD, and ICDE conferences, and has been awarded 20+ granted and pending patents. He received his Masters degree from the Media Laboratory, Massachusetts Institute of Technology, and holds a Bachelor degree from the department of Computer Science, Indian Institute of Technology, Kharagpur.

Ippokratis Pandis is a principal engineer at Amazon Web Services working on AWS Redshift. AWS Redshift is Amazon's fully managed, petabyte scale data warehouse service. Previously, Ippokratis has held positions as software engineer at Cloudera where he worked on the Impala SQL-on-Hadoop query engine and as member of the research staff at IBM Almaden Research Center. At IBM, he was member of the team that designed and implemented the BLU column-store engine, which currently ships as part of IBM's DB2 LUW v10.5 with BLU Acceleration. Ippokratis received his PhD from Carnegie Mellon University. He is the recipient of best demonstration and paper awards at ICDE 2006, SIGMOD 2011 and CIDR 2013. He has also served as PC chair of DaMoN 2014, DaMoN 2015 and CloudDM 2016.

Rajkumar Sen is a Software Development Director at Oracle Inc. responsible for architecting Oracle's Business Intelligence Analytics for the Cloud. Prior to that, he was a Director, Engineering at MemSQL Inc. where he architected the query optimizer and the distributed query processing engine, Principal Engineer at Oracle Inc. where he developed features for the Oracle database query optimizer, and Senior Staff Engineer at Sybase Inc. where he architected the Distributed Object Lock Manager and Metadata Manager for Sybase ASE Cluster Edition. He received his Masters Degree in Computer Science with specialization in Databases from Indian Institute of Technology, Mumbai, India in 2004 and his current research interests are in the areas of query optimization and distributed query processing.

8. REFERENCES

[1] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: A new model and architecture for data stream management. *The VLDB Journal*, 12(2), 2003.

[2] D. J. Abadi, D. S. Myers, D. J. DeWitt, and S. R. Madden. Materialization Strategies in a Column-Oriented DBMS. In *ICDE*, 2007.

[3] A. Ailamaki, D. J. DeWitt, and M. D. Hill. Data page layouts for relational databases on deep memory hierarchies. *VLDB J.*, 11(3):198–215, 2002.

[4] D. S. Batory. On searching transposed files. *ACM Trans. Database Syst.*, 4(4):531–544, Dec. 1979.

[5] G. Candea, N. Polyzotis, and R. Vingralek. A Scalable, Predictable Join Operator for Highly Concurrent Data Warehouses. *PVLDB*, 2(1), Aug. 2009.

[6] R. L. Cole, F. Funke, L. Giakoumakis, W. Guy, A. Kemper, S. Krompass, H. A. Kuno, R. O. Nambiar, T. Neumann, M. Poess, K. Sattler, M. Seibold, E. Simon, and F. Waas. The mixed workload CH-benCHmark. In *DBTest*, 2011.

[7] G. P. Copeland and S. N. Khoshafian. A decomposition storage model. *SIGMOD Rec.*, 14(4):268–279, May 1985.

[8] J. Dittrich and A. Jindal. Towards a one size fits all database architecture. In *CIDR*, 2011.

[9] G. Giannikis, G. Alonso, and D. Kossmann. SharedDB: Killing one thousand queries with one stone. *PVLDB*, 5(6), 2012.

[10] G. Giannikis, D. Makreshanski, G. Alonso, and D. Kossmann. Shared workload optimization. *PVLDB*, 7(6), 2014.

[11] R. A. Hankins and J. M. Patel. Data Morphing: An Adaptive, Cache-Conscious Storage Technique. In *VLDB*, 2003.

[12] S. Harizopoulos, V. Shkapenyuk, and A. Ailamaki. QPipe: A Simultaneously Pipelined Relational Query Engine. SIGMOD, 2005.

[13] Y. He, R. Lee, Y. Huai, Z. Shao, N. Jain, X. Zhang, and Z. Xu. Rcfile: A fast and space-efficient data placement structure in mapreduce-based warehouse systems. In *ICDE*, 2011.

[14] S. Héman, M. Zukowski, N. J. Nes, L. Sidirourgos, and P. Boncz. Positional Update Handling in Column Stores. SIGMOD, 2010.

[15] A. L. Holloway, V. Raman, G. Swart, and D. J. DeWitt. How to Barter Bits for Chronons: Compression and Bandwidth Trade Offs for Database Scans. SIGMOD, 2007.

[16] H. V. Jagadish, P. P. S. Narayan, S. Seshadri, S. Sudarshan, and R. Kanneganti. Incremental Organization for Data Recording and Warehousing. In *VLDB*, 1997.

[17] A. Jindal, E. Palatinus, V. Pavlov, and J. Dittrich. A Comparison of Knives for Bread Slicing. *PVLDB*, 6(6):361–372, 2013.

[18] A. Jindal, J.-A. Quiané-Ruiz, and J. Dittrich. Trojan data layouts: right shoes for a running elephant. In *SoCC*, 2011.

[19] A. Kemper and T. Neumann. HyPer: A hybrid OLTP / OLAP main memory database system based on virtual memory snapshots. In *ICDE*, 2011.

[20] M. Kornacker, A. Behm, V. Bittorf, T. Bobrovytsky, C. Ching, A. Choi, J. Erickson, M. Grund, D. Hecht, M. Jacobs, I. Joshi, L. Kuff, D. Kumar, A. Leblang, N. Li, I. Pandis, H. Robinson, D. Rorke, S. Rus, J. Russell, D. Tsirogiannis, S. Wanderman-Milne, and M. Yoder. Impala: A modern, open-source SQL engine for Hadoop. In *CIDR*, 2015.

[21] S. Kulkarni, N. Bhagat, M. Fu, V. Kedigehalli, C. Kellogg, S. Mittal, J. M. Patel, K. Ramasamy, and S. Taneja. Twitter Heron: Stream processing at scale. In *SIGMOD*, 2015.

[22] T. Lahiri, S. Chavan, M. Colgan, D. Das, A. Ganesh, M. Gleeson, S. Hase, A. Holloway, J. Kamp, T.-H. Lee, J. Loaiza, N. MacNaughton, V. Marwah, N. Mukherjee, A. Mullick, S. Muthulingam, V. Raja, M. Roth, E. Soylemez, and M. Zat. Oracle database in-memory: A dual format in-memory database. *ICDE*, 2015.

[23] Y. Li, I. Pandis, R. Mueller, V. Raman, and G. Lohman. NUMA-aware algorithms: the case of data shuffling. In *CIDR*, 2013.

[24] T. Lipcon, D. Alves, D. Burkert, J.-D. Cryans, A. Dembo, M. Percy, S. Rus, W. Dave, M. Bertozzi, C. P. McCabe, and A. Wang. Kudu: Storage for fast analytics on fast data. `http://getkudu.io/kudu.pdf`.

[25] J. Meehan, N. Tatbul, S. Zdonik, C. Aslantas, U. Cetintemel, J. Du, T. Kraska, S. Madden, D. Maier, A. Pavlo, M. Stonebraker, K. Tufte, and H. Wang. S-Store: Streaming Meets Transaction Processing. *PVLDB*, 8(13), 2015.

[26] Memsql skip list. http://blog.memsql.com/the-story-behind-memsqls-skiplist-indexes/.

[27] N. Mukherjee, S. Chavan, M. Colgan, D. Das, M. Gleeson, S. Hase, A. Holloway, H. Jin, J. Kamp, K. Kulkarni, T. Lahiri, J. Loaiza, N. MacNaughton, V. Marwah, A. Mullick, A. Witkowski, J. Yan, and M. Zat. Distributed architecture of oracle database in-memory. *PVLDB*, 8(12), 2015.

[28] T. Neumann. Efficiently Compiling Efficient Query Plans for Modern Hardware. *PVLDB*, 4(9), 2011.

[29] P. E. O'Neil, E. Cheng, D. Gawlick, and E. J. O'Neil. The Log-Structured Merge-Tree (LSM-Tree). *Acta Inf.*, 33(4), 1996.

[30] H. Plattner. A Common Database Approach for OLTP and OLAP Using an In-memory Column Database. In *SIGMOD*, 2009.

[31] I. Psaroudakis, T. Scheuer, N. May, A. Sellami, and A. Ailamaki. Scaling up concurrent main-memory column-store scans: Towards adaptive numa-aware data and task placement. *PVLDB*, 8(12):1442–1453, 2015.

[32] I. Psaroudakis, F. Wolf, N. May, T. Neumann, A. Böhm, A. Ailamaki, and K. Sattler. Scaling up mixed workloads: A battle of data freshness, flexibility, and scheduling. In *TPCTC*, 2014.

[33] R. Ramamurthy, D. J. DeWitt, and Q. Su. A Case for Fractured Mirrors. In *VLDB*, 2002.

[34] V. Raman, G. Attaluri, R. Barber, N. Chainani, D. Kalmuk, V. KulandaiSamy, J. Leenstra, S. Lightstone, S. Liu, G. M. Lohman, T. Malkemus, R. Mueller, I. Pandis, B. Schiefer, D. Sharpe, R. Sidle, A. Storm, and L. Zhang. DB2 with BLU Acceleration: So much more than just a column store. *PVLDB*, 6, 2013.

[35] V. Sikka, F. Färber, W. Lehner, S. K. Cha, T. Peh, and C. Bornhövd. Efficient transaction processing in SAP HANA database: the end of a column store myth. In *SIGMOD*, 2012.

[36] A. Skidanov, A. Papito, and A. Prout. A Column Store Engine for Real-Time Streaming Analytics. In *ICDE*, 2016.

[37] M. Stonebraker and U. Cetintemel. "One Size Fits All": An Idea Whose Time Has Come and Gone (Abstract). In *ICDE*, 2005.

[38] M. Stonebraker et al. The End of an Architectural Era (It's Time for a Complete Rewrite). In *VLDB*, 2007.

[39] P. Unterbrunner, G. Giannikis, G. Alonso, D. Fauser, and D. Kossmann. Predictable Performance for Unpredictable Workloads. *PVLDB*, 2(1), 2009.

[40] S. D. Viglas. Just-in-time compilation for SQL query processing. In *ICDE*, 2014.

[41] S. Wanderman-Milne and N. Li. Runtime code generation in Cloudera Impala. *IEEE Data Eng. Bull.*, 2014.

[42] T. Willhalm, N. Popovici, Y. Boshmaf, H. Plattner, A. Zeier, and J. Schaffner. SIMD-scan: Ultra Fast In-memory Table Scan Using On-chip Vector Processing Units. *PVLDB*, 2(1), Aug. 2009.