1 Column-store Technology for Read Intensive Workloads

1.1 Introduction

The traditional paradigm of row-oriented database systems has faced several performance barriers as businesses and research facilities are performing more read-intensive workloads. As CPU processing speeds have continued to rise, IO performance remains a critical bottleneck for many database systems [5]. One proposed solution for improving read performance is a new architectural design referred to as column-oriented database systems, or "column-store" for simplicity. Column-store, in a nutshell, stores the attribute values column-by-column onto disk pages; row store, on the other hand, stores tables tuple-by-tuple onto disk pages. The ability to read only the columns of interest from a database has piqued the interest of data warehouse, decision support, scientific data-mining and business intelligence application developers due to their read-tuned storage layout [1, 6].

This section aims to accomplish three things: 1) to highlight the primary components of column-store technology, 2) to provide a critical analysis of commercial column-store technologies, and 3) to explore the future of hybrid column row technology. By understanding the clear benefits of column-store databases, we may make more practical decisions concerning database design in future situations. Later, a critique of column-store performance will be provided, including a look at hybrid column-row stores that attempt to unify the advantages of both architectures.

1.2 History and Motivation

Before column-store, the NSM storage model, or nary storage model, was the storage model for row-oriented databases. NSM stored all n attributes of a record together on disk, and were generally considered to be the best approach for database systems [9]. Motivation towards column-oriented databases began in the 1970s when medical researchers began using statistical analysis on patient records to improve care decisions [8]. Searching for improved read performance, the medical researchers worked with computer scientists to develop a database bank for patient records that included "transposed" auxiliary data files for efficient retrieval [8].

10 years later a new model titled the Decomposition Storage Model, or DSM for short, was developed as an alternative to NSM by storing attribute values in clusters on disk and using "surrogates", or primary keys, to explicitly represent a database entity [9]. While this model required twice as many writes for updates compared to NSM, the ability to compress column attributes as well as improved cache performance distinguished it from NSM [9]. DSM introduced researchers to the advantages of vertically partitioned databases, sparking new research towards ways of improving tuple reconstruction and update performance in DSM databases, even at the expense of adding CPU overhead [10, 11].

1.3 Overview of Column-store Architecture

Column-store databases are unique in the way data is stored column-by-column on disk, allowing queries to only touch the columns necessary to fulfill a query. Built for read-intensive workloads,

column-store databases are ideal for read-only queries that touch few columns from large tables with wide tuples. Row-stores have benefitted from compression techniques such as Dictionary Encoding in reducing the size of tuples on disk, yet the high degree of difference between adjacent values in a tuple makes it difficult to efficiently compress tuples [12]. One of the major benefits of column-store databases is the ability to compress page data at a high ratio. This reduces the distance between data on disk and reduces the amount of data passed to memory, improving overall system throughput [12]. Compressing data provides more space for the database system to store copies in different sort orders, providing the query optimizer with more access-plan options [5]. While compression and decompression require additional CPU overhead, the trade-off for reduced IO time has been found to outweigh the cost of decompression and improve performance [11].

1.4 Application

Imagine a retail company has a 10 million row, 100 column fact table that stores all the transactions from different stores, and they have business analysts who want to get information such as how many of product X were purchased in comparison to Product Y, or the total number of purchases above 100 $. Now image the queries only need data from 10 of the 100 columns to execute the query. In a row-store, we would have to read in all attribute values for a tuple, wasting disk bandwidth on irrelevant data. In a column-store, the amount of irrelevant data read in is minimized by only reading the column data relevant to the query.

The column-store architecture have clear commercial value as more and more enterprises entered the data-mining and business analysis markets [4]. Data centers and corporations are interested in improving the efficiency of read scans because they run data-warehousing and business intelligence applications that read large fact-tables to gather analytics [1]. As previously noted, medical researchers often depend on data analysis to support diagnosis and treatment decisions [8].

1.5 Vendor Analysis

Vendors such as Sybase IQ, MonetDB, VectorWise, Vertica, and Kx Systems have developed different approaches towards implementing column-store databases. Many of these vendors claim to be built "from the ground up", with some boasting benchmark results showing an impressive 270x query response time speedup [11, 13]. They primarily targeting business intelligence and decision support applications, focusing on providing low latency analytics on massive datasets [11, 13, 14, 15]. The following is an overview of each vendor that pays special attention to the unique aspects of each product.

1.5.1 SybaseIQ

Sybase IQ was one of the first commercial column-store database systems designed for analytics, providing true column-oriented storage [17]. Sybase IQ maximizes disk bandwidth by increasing page size and compressing data in order to fit as much data onto a page as possible [17]. SybaseIQ columns can have multiple indexes because data is amortized through compression, allowing the

query executor a larger search space for calculating access plans. SybaseIQ compresses chooses the best compression scheme for the database, and compresses the index records into segmented bitmaps used by the query engine to quickly resolve where clauses using boolean operators [17].

1.5.2 MonetDB

MonetDB is a vertically fragmented database that focuses on improving CPU efficiency by taking advantage of optimizations such as loop pipelining [11]. MonetDB's key characteristics are its use of lightweight data compression, late materialization, and vector processing [11, 18]. MonetDB targets IPC efficiency, marking query expression calculation and lack of loop pipeline as major performance detriments in row-store database systems [11]. Columns are represented on disk using Binary Association Tables (BAT), which are two-column tables with the first column representing the oid, or object id, and the second column is the attribute value [11]. The X100 Vectorized Query Processor is MonetDB query engine that is cache-conscious, storing data vertically in the cache in order to take advantage of the loop-pipelining compiler optimizations [11].

1.5.3 Vertica

Vertica is the commercial version of the open-source C-Store database [site]. Compared to MonetDB and SybaseIQ, Vertica supports hybrid-store architecture that contains a main-memory write-store optimized for updates and inserts and a read-store optimized for ad-hoc queries [5, 16]. Both the write and read store are column-oriented, allowing for the stores to share one column-oriented query executor. The write-store moves tuples asynchronously to the read-store, providing for decent update and insert performance [18]. In their technical overview, Vertica claims to have an "aggressive" column compression scheme that on average can achieve a 90% reduction, but clearly this depends on the type of data stored, its sort order and the number of distinct values it has (cardinality) [16]. Vertica samples column-data in order to determine the best compression scheme in order to provide enough room for multiple "projections", or copies of the column data in different sort orders [13, 16]. Like Sybase, Vertica uses multiple projections to optimize query execution [17, 18].

1.5.4 VectorWise

VectorWise claims to be "unique" because it takes full advantage of CPU performance features overlooked by competitors such as using SIMD (single instruction, multiple data) instructions to process vectorized data, however other column-stores like MonetDB provide similar SIMD support [15, 18]. In order to increase system throughput, VectorWise isolates all CPUs and caches to avoid memory contention [15]. VectorWise also provides support for row-oriented tables as an disk-conservative alternative for wide-tables with few rows, which can be beneficial for database administrators who want to further tune a VectorWise database [15]. Similar to Vertica, VectorWise includes an in-memory data structure for supporting efficient updates and inserts, called a Positional Delta Tree [15]. The amount of memory allocated for the PDT can also be tuned for performance.

Unlike other column-oriented databases, VectorWise supports the use of storage indexes, which provide the minimum and maximum values of each page that helps avoid reading unnecessary pages from disk [15].

1.5.5 Kx Systems

Kx Systems have designed their column-store database kdb+ to gather and mine both historical (disk) and real-time (RAM) data efficiently, ideal for investment traders and risk management [14]. Handling real-time data in a column-store is a challenge due to its suboptimal insert and updates architecture. In order to provide real-time performance, kdb+ maintains an in-memory database of new data As well, Kx Systems is built to support high parallelism across distributed databases, providing core support for UUIDs as well as a publish and subscription mechanism for load-balancing [14]. Unlike other competitors, Kx Systems provides their own programming language "q", a vector processing language intended to compliment a column-oriented database [14].

1.6 Criticisms

Researchers have been picking apart the components of classic column-store databases such as MonetDB and C-store in order to determine whether the benefits of column-store can be implemented in a row-store architecture with minimal consequences [1, 3]. Clearly, one can expect column-oriented databases to perform poorly at updating multiple columns of a tuple or at reading full tuples from the database, yet experiments in batch updating has aimed to improve column-store update performance in multi-core environments [7].

1.7 Analysis

Column-stores are a domain-specific technology, meaning they provide a specific solution to a specific type of problem, in this case, read-intensive queries on massive datasets. The consequence of being domain-specific is that the technology is limited outside of its intended domain. Domain-specific technologies are also capable of losing their niche once general-purpose technologies begin targeting their domain of interest. Column-stores are clearly powerful at executing ad-hoc queries, and in the hands of powerful analysts can become powerful assets for driving business decisions. That said, unless you are in the position of needing to develop a database system for intensive read-heavy workloads and minimal inserts and updates, then column-stores will be a poor choice.

Right now, "Big Data" and "Data Mining" are massively popular frontiers in the computer science world today, spurring researchers to find better ways to get more information out of massive databases.

1.8 Bibliography

1 Harizopoulos, P̈erformance Tradeoffs in Read-Optimized Databases2̈ Larson C̈olumnar Storage in SQL Server 20123̈ Holloway R̈ead-Optimized Databases, In Depth4̈ Holsheimer Ärchitectural support for data mining5̈ Stonebraker, Abadi C̈-Store: A Column-oriented DBMS6̈ Abadi C̈olumn-oriented Database Systems7̈ Krueger F̈ast Updates on Read-Optimized Databases Using Multi-Core CPUs8̈ Weyl Ä modular Self-Describing Clinical Databank System9̈ Copeland Ä Decomposition Storage Model1̈0 Ramamurthy Ä Case for Fractured Mirrors1̈1 Boncz M̈onetDB/X100: Hyper-Pipelining Query Execution1̈2 Abadi Ïntegrating Compression and Execution in Column-Oriented Database Systems1̈3 Vertica White Paper 14 Kx Solutions White Paper 15 VectorWise White Paper 16 Vertica White Paper 17 MacNicol "Sybase IQ Multiplex Designed For Analytics" 18 VLDB 2009 Tutorial 19 The Vertica Analytic Database 20 kx solutions trading and risk management