Tutorial: SQL-on-Hadoop Systems

Daniel Abadi
Yale University
daniel.abadi@yale.edu

Shivnath Babu
Duke University
shivnath@cs.duke.edu

Fatma Özcan
IBM Research - Almaden
fozcan@us.ibm.com

Ippokratis Pandis
Cloudera
ippokratis@cloudera.com

1. INTRODUCTION

Enterprises are increasingly using Apache Hadoop, more specifically HDFS, as a central repository for all their data; data coming from various sources, including operational systems, social media and the web, sensors and smart devices, as well as their applications. At the same time many enterprise data management tools (e.g. from SAP ERP and SAS to Tableau) rely on SQL and many enterprise users are familiar and comfortable with SQL. As a result, SQL processing over Hadoop data has gained significant traction over the recent years, and the number of systems that provide such capability has increased significantly. In this tutorial we use the term SQL-on-Hadoop to refer to systems that provide some level of declarative SQL(-like) processing over HDFS and noSQL data sources, using architectures that include computational or storage engines compatible with Apache Hadoop.

It is important to note that there are important distinct characteristics of this emerging eco-system that are different than traditional relational warehouses. First, in the world of Hadoop and HDFS data, complex data types, such as arrays, maps, structs, as well as JSON data are more prevalent. Second, the users utilize UDFs (user-defined-functions) very widely to express their business logic, which is sometimes very awkward to express in SQL itself. Third, often times there is little control over HDFS. Files can be added or modified outside the tight control of a query engine, making statistics maintenance a challenge. These factors complicate the query optimization further in the Hadoop system.

There is a wide variety of solutions, system architectures, and capabilities in this space, with varying degree of SQL support and capabilities. The purpose of this tutorial is to provide an overview of these options, discuss various different approaches, and compare them to gain insights into open research problems.

In this tutorial, we will examine the SQL-on-Hadoop systems along various dimensions. One important aspect is their data storage. Some of these systems support all native Hadoop formats, and do not impose any propriety data formats, and keep the data open to all applications running on the same platform. While there are some database hybrid solutions, such as HAWQ, HP Haven, and Vortex, that store their propriety data formats in HDFS. Most often, these systems are also able to run SQL queries over native HDFS formats, but do not provide the same level of performance.

Some SQL-on-Hadoop systems provide their own SQL-specific run-times, such as Impala, Big SQL, and Presto, while others exploit a general purpose run-time such as Hive (MapReduce and Tez) and SparkSQL (Spark).

Another important aspect is the support for schema flexibility and complex data types. Almost all of these systems support complex data types, such as arrays and structs. But, only a few, such as Drill and Hadapt with Sinew [13], are able to work with schemaless data.

Proceedings of the VLDB Endowment, Vol. 8, No. 12

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, and discuss several research problems they address.

At the end, we will wrap up by summarizing all architectures, their pros, and cons, and how we see this space evolving, and where we think more research is needed. We expect this tutorial to be the engaging enough for the audience and act as a bridge to start a lively open discussion.

3. OVERVIEW OF REPRESENTATIVE SYSTEMS

Hive [14], an open source project originally built at Facebook, was the first SQL-on-Hadoop offering that provided an SQL-like query language, called HiveQL, and used MapReduce run-time to execute queries. Hive compiled HiveQL queries into a series of map reduce jobs. As SQL-on-Hadoop gained popularity, MapReduce based run-time did not provide the required response times, due to the high latency in launching map reduce jobs. To address this issue, Hive moved to a different run-time, Tez [10], which can run DAGs as a single job, reducing the latency in launching jobs. Meanwhile, the Facebook team developed a second SQL-on-Hadoop offering, called Presto, that uses a traditional MPP DBMS run-time instead of MapReduce.

Hadapt, which spun out of the HadoopDB research project [1], was the first commercial SQL-on-Hadoop offering. Hadapt and HadoopDB replaced the file-oriented HDFS storage formats with DBMS-oriented storage, including column-store

data layouts. Hadapt leverages two different query run-times: a MapReduce-based run-time (like the original Hive run-time) for long queries that require fault tolerance, and an interactive MPP run-time for shorter queries.

Spark is a fast, general purpose cluster computing engine that is compatible with Hadoop data and tries to address the shortcomings of MapReduce. There are three different systems that use Spark as their run-time for SQL processing: Shark [16], Hive on Spark [8], and Spark SQL [2].

As it became clear that the latency of launching jobs for each query is too expensive for interactive SQL query processing, there was a shift to shared-nothing database architectures for SQL processing over Hadoop data. Hadapt, Impala, Presto, Drill, as well as Big SQL all employ such MPP architectures, where a long-running process co-exists with DataNodes on each node in the cluster, and continuously answers SQL queries.

Cloudera Impala [9] is an open-source, fully-integrated MPP SQL query engine. Unlike other systems (often forks of Postgres), Impala is a brand-new engine. Impala reads at almost disk bandwidth and is typically able to saturate all available disks. A main characteristic of Impala is that employs LLVM to generate code at runtime to speed up frequently executed code paths [15].

IBM Big SQL [7] leverages IBM's state-of-the-art relational database technology, to processes standard SQL queries over HDFS data, supporting all common Hadoop file formats, without introducing any propriety formats. Big SQL 3.0 shares the same catalog and table definitions with Hive using the Hive Metastore. Big SQL exploits sophisticated query rewrite transformations [11, 17] that are targeted for complex nested decision support queries. It uses sophisticated data statistics and a cost-based optimizer to choose the best query execution plan. Big SQL introduces a scheduler service that assigns HDFS blocks to database workers for processing on a query by query basis.

Apache Drill [3] is an open-source project which aims at providing SQL-like declarative processing over self-describing semi-structured data. Its focus is on analyzing data without imposing a fixed schema or creating tables in a catalog like Hive MetaStore. It runs queries over files and HBase tables, and discovers data when reading input data. For every fixed chunk of data, it discovers its schema, creates an in-memory columnar representation, and generates specific code for processing. As such, it can accommodate data chunks with varying schemas.

Several SQL-on-Hadoop systems leverage existing relational database technology: HadoopDB [1] uses large amounts of PostgreSQL code; HAWQ [6] uses large amounts of Greenplum code; and Vortex [5] uses large amounts of Actian Vectorwise code. In some cases, database files are stored in HDFS, while in other cases, database files are stored on the same physical machines as HDFS, but on a separate file system. In some cases, data is dynamically moved from Hadoop file formats to the native storage structures of the DBMS. In some cases, queries are executed by the database engine code, while in other cases, query execution is split between database engine code and native Hadoop execution engines such as MapReduce or Tez [4].

An important category of SQL-on-Hadoop includes systems that provide some level of SQL support over HBase data. HBase provides auto-sharding and fail over technology for scaling tables across multiple servers. It also enables updates, running on top of the HDFS, which itself does not support updates. HBase scales out to petabytes of data easily over a cluster of commodity hardware.

Splice Machine [12] provides SQL support over HBase data using Apache Derby, targeting both operational as well as analytical workloads. It replaces the storage system and run-time of Derby with HBase and Hadoop. Splice Machine leverages the Derby compiler stack to generate execution plans that access HBase servers. (like SQL stored procedures) which enables pushing computations down to each region (shard) of HBase.

Phoenix provides SQL querying over HBase via an embeddable JDBC driver built for high performance and read/write operations. It converts SQL queries into execution plans composed of HBase scans. Coprocessors and custom filters are leveraged in order to improve performance. Phoenix provides secondary indexes as well as basic support for joins, both of which are difficult to get with HBase.

4. REFERENCES

[1] A. Abouzeid, K. Bajda-Pawlikowski, D. J. Abadi, A. Rasin, and A. Silberschatz. HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. *PVLDB*, 2009.

[2] M. Amburst, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, and M. Zaharia. Spark SQL: Relational data processing in Spark. In *ACM SIGMOD*, 2015.

[3] Apache Drill. `http://drill.apache.org/`.

[4] K. Bajda-Pawlikowski, D. J. Abadi, A. Silberschatz, and E. Paulson. Efficient processing of data warehousing queries in a split execution environment. In *SIGMOD*, 2011.

[5] P. Boncz. Vortex: Vectorwise goes Hadoop. http://databasearchitects.blogspot.com/2014/05/vectorwise-goes-hadoop.html.

[6] L. Chang, Z. Wang, T. Ma, L. Jian, L. Ma, A. Goldshuv, L. Lonergan, J. Cohen, C. Welton, G. Sherry, and M. Bhandarkar. HAWQ: A massively parallel processing SQL engine in hadoop. In *SIGMOD*, 2014.

[7] S. Gray, F. Özcan, H. Pereyra, B. van der Linden, and A. Zubiri. IBM Big SQL 3.0: SQL-on-Hadoop without compromise. `http://public.dhe.ibm.com/common/ssi/ecm/en/sww14019usen/SWW14019USEN.PDF`, 2014.

[8] Hive on spark. `https://cwiki.apache.org/confluence/display/Hive/Hive+on+Spark`.

[9] M. Kornacker and et.al. Impala: A modern, open-source SQL engine for Hadoop. In *CIDR*, 2015.

[10] B. Saha, H. Shah, S. Seth, G. Vijayaraghavan, A. Murthy, and C. Curino. Apache Tez: A unifying framework for modeling and building data processing applications. In *SIGMOD*, 2015.

[11] P. Seshadri, H. Pirahesh, and T. Y. C. Leung. Complex query decorrelation. In *ICDE*, 1996.

[12] Splice machine. `http://www.splicemachine.com/`.

[13] D. Tahara, T. Diamond, and D. J. Abadi. Sinew: A SQL System for Multi-structured Data. In *ACM SIGMOD*, 2014.

[14] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, N. Zhang, S. Anthony, H. Liu, and R. Murthy. Hive - A Petabyte Scale Data Warehouse Using Hadoop. In *ICDE*, 2010.

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

[16] R. S. Xin, J. Rosen, M. Zaharia, M. J. Franklin, S. Shenker, and I. Stoica. Shark: SQL and rich analytics at scale. In *ACM SIGMOD*, 2013.

[17] C. Zuzarte, H. Pirahesh, W. Ma, Q. Cheng, L. Liu, and K. Wong. WinMagic : Subquery elimination using window aggregation. In *ACM SIGMOD*, 2003.