A brief comparative perspective on SQL access for Hadoop

Sherif Sakr
Senior Research Scientist
National ICT Australia

12 August 2014

Although Hadoop is often thought of as the one-size-fits-all solution for big data processing problems, the project is limited in its ability to manage large-scale graph processing, stream processing, and scalable processing of structured data. Learn about Big SQL, a massively parallel processing SQL engine that is optimized for processing large-scale structured data. See how it compares to other systems that were recently introduced to improve the efficiency of the Hadoop framework for processing large-scale structured data.

Why SQL access for Hadoop?

The growing demand for large-scale data processing and data analysis applications spurred the development of novel solutions from the industry (for example, web data analysis, click-stream analysis, network-monitoring log analysis, social network data analysis, and enterprise data analysis) and the sciences (for example, analysis of data that is produced by massive-scale simulations, sensor deployments, and high-throughput lab equipment). The Hadoop framework provides a simple but powerful programming model and runtime environment that eases the job of developing scalable parallel applications to process vast amounts of data on large clusters of commodity machines.

As an open source project, Hadoop is applied to many application domains in academia and industry. IBM®, Oracle, Microsoft, and several successful startups such as Cloudera, MapR, Platfora, and Trifecta have built solutions that are based on Hadoop. At some point, the words *Hadoop* and *big data* came to be used interchangeably. Gartner estimates that the current Hadoop ecosystem market is worth roughly US$77 million and this market size is expected to reach US $813 million by 2016.

Recently, academia and industry have started to recognize the limitations of the Hadoop framework in several application domains. (Several of these limitations are described in this article. Learn more about limitations of the Hadoop framework in this industry report. See more reports in Resources.) For example, in the domain of large-scale graph processing, the Hadoop framework is shown to be inefficient. Google recently introduced the Pregel system, open-sourced by Apache Giraph and Apache Hama projects, that uses a bulk synchronous parallel (BSP) based programming model for efficient and scalable processing of massive graphs on distributed clusters of commodity machines. Several other projects (for example, GraphLab and GraphChi) were introduced to tackle the problem of large-scale graph processing.

In large-scale processing of streaming data, Hadoop seems to be an inadequate platform. Twitter announced the release of the Storm system that fills this gap by providing a distributed and fault-tolerant platform for implementing continuous and real-time processing applications of streamed data. Other systems in this domain include IBM InfoSphere Streams and Apache S4.

In processing large-scale structured data, several studies reported the significant inefficiency of the Hadoop framework. In particular, the studies claim that Hadoop is the wrong choice for interactive queries that have a target response time of a few seconds or milliseconds.

Another drawback is that programmers in this domain are likely to be unfamiliar with the MapReduce framework. In practice, the Hadoop framework requires a developer to engage in a steep learning curve and apply strong programming expertise. Many programmers prefer to use SQL (in which they are more proficient), which enables them to express tasks in a high-level declarative language and leave the optimization details to the backend engine. High-level language abstractions enable the underlying system to perform automatic optimization. This article focuses on the new generation of big data systems that aim to provide efficient and scalable engines for processing massive, structured data. The article starts with an overview of the massively parallel processing SQL engine, Big SQL. It compares Big SQL to similar systems: Apache HIVE, Cloudera Impala, and Pivotal HAWQ.

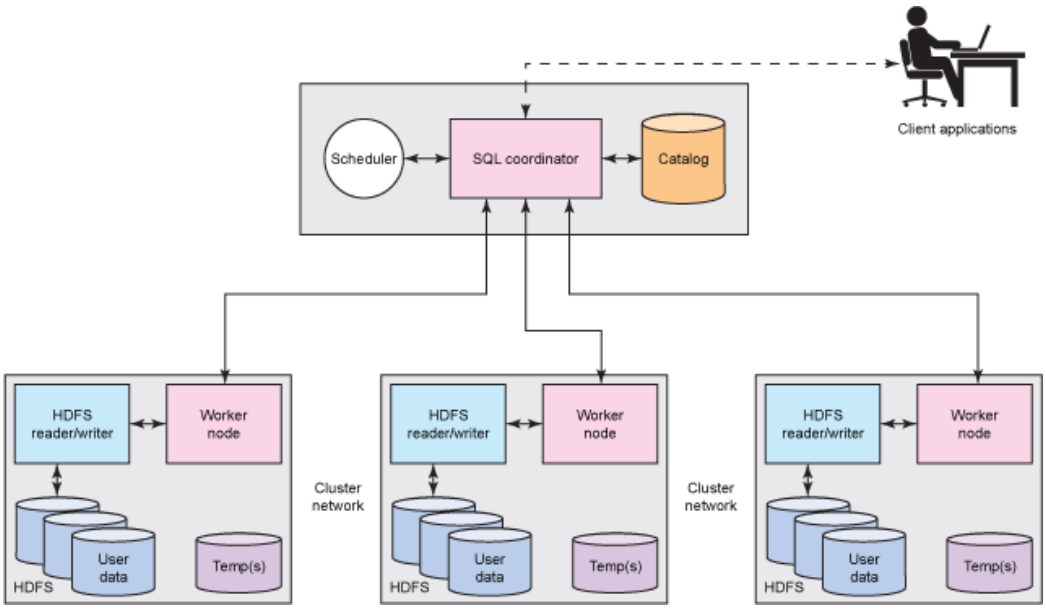
Introduction to Big SQL

Big SQL is the SQL interface for the IBM big data processing platform, InfoSphere BigInsights, which is built on the Apache Hadoop framework. In particular, it provides SQL access to data that is stored in InfoSphere BigInsights and uses the Hadoop framework for complex data sets and direct access for smaller queries. In the initial implementation of Big SQL, the engine decomposes the SQL query into a series of Hadoop jobs. For interactive queries, Big SQL relies on a built-in query optimizer that rewrites the input query as a local job to help minimize latencies by using Hadoop dynamic scheduling mechanisms. The query optimizer also takes care of traditional query optimization such as *optimal order*, in which tables are accessed and the most efficient `join` strategy implements the query.

The design of the recent version of the Big SQL engine is different. It applies a shared-nothing parallel database architecture, in which it replaces the underlying Hadoop framework with a massively parallel processing SQL engine that is deployed directly on the physical Hadoop Distributed File System (HDFS). Therefore, Big SQL remains a harmonious member of the Hadoop ecosystem because all data is stored in its original HDFS-based format rather than in a propriety storage format. The data can be accessed by all other tools of Hadoop ecosystem, such as Pig and Hive. The system infrastructure provides a logical view of the data through storage and management of metadata information. In particular, a *table* is simply a view that is defined over the stored data in the underlying HDFS. In addition, the Big SQL engine uses the Apache Hive database catalog facility for storing the information about table definitions, location, and storage format.

Figure 1 illustrates the general architecture of the Big SQL engine. In this architecture, the SQL coordinator node is responsible for compiling and optimizing the input SQL statements to generate an efficient, parallel plan for how to run queries. This plan is distributed among a set of worker nodes to run. These worker nodes are continuously running so that there is no startup latency.

Figure 1. Architecture of the Big SQL engine



Based on available resources, the Big SQL engine dynamically and automatically determines the best degree of intra-query parallelism per query. The scheduler component of the Big SQL engine uses the catalog information about the location of the data to ensure that work is processed efficiently, as close to the data as possible. The scheduler component also conducts further optimizations on the query plan by understanding the physical layout of the data and associated statistics. For example, it can decide to eliminate partitions that are not relevant for a query predicate. The Big SQL runtime engine provides in-memory caching capabilities and includes the ability to spill large data sets to the local disk at each node that is processing a query. Therefore, the only limit to the size of the queries, groups, and sorting is the disk capacity of the cluster.

In practice, Big SQL doesn't run in isolation. It is common that nodes are shared with other tools and jobs in the Hadoop ecosystem, for example, MapReduce jobs, HBase nodes, and Pig jobs. In this way, Big SQL can be configured to limit its footprint on the cluster resources (for example, the percentage of processor use and the percentage of memory use). Configuration settings can specify that resources get automatically adjusted based on workload.

With Big SQL, you also can query heterogeneous systems, as in the case where you need to join data that is stored in HDFS with data that is stored in other relational databases such as IBM® DB2®, Oracle, and Teradata. Big SQL supports open integration across business analytic tools such as IBM® Cognos® and Microstrategy.

Big SQL supports a rich SQL dialect that extends the `SQL:2011` language and supports broad relational data types and supports the use of stored procedures and user-defined functions. It also provides full support for the following operations:

- SQL subquery mechanisms:
  - `EXISTS`
  - `NOT`
  - `EXISTS`
  - `IN`
  - `ANY`
  - `SOME`
  - `HAVING`
- Standard `join` operations:
  - `inner`
  - `outer`
  - `equality`
  - `non-equality`
- Set manipulating operations:
  - `union`
  - `intersect`
  - `except`
- Extensive analytic capabilities:
  - Grouping sets with `CUBE`
  - Grouping sets with `ROLLUP`
  - Standard analytic functions:
    - `CORRELATION`
    - `COVARIANCE`
    - `STDDEV`

- `VARIANCE`

Comparison with other big SQL systems

Several systems also give SQL support to Hadoop. This article includes a brief overview of some of the key systems and compares them to IBM Big SQL.

Hive

Apache Hive is considered to be the first to support SQL-on-Hadoop. This open source data warehousing solution is built on the Hadoop platform. It supports familiar relational database concepts such as tables, columns, and partitions and includes some SQL support for unstructured data. Hive maintains the extensibility and flexibility that Hadoop offers. Hive supports all of the major primitive types (for example, `integers`, `floats`, and `strings`) and complex types (for example, `maps`, `lists`, and `structs`). It also supports queries that are expressed in an SQL-like declarative language, Hive Query Language (HiveQL), which represents a subset of SQL92, and therefore can be easily understood by anyone who is familiar with SQL. These queries automatically compile into MapReduce jobs that are run by using Hadoop. HiveQL enables users to plug custom MapReduce scripts into queries.

In general, Hive is a good interface for anyone from the relational database world, even though the details of the underlying implementation aren't hidden. HiveQL does differ from standard SQL in some ways, such as the best way to specify `join` operations for best performance. HiveQL is missing some language features. It does offer the ability to plug in custom code for situations that don't fit into SQL, and it includes tools to handle input and output. Hive lacks support for the following operations:

- `UPDATE` or `DELETE` statements
- `INSERT` operations for single rows
- Date or time data types, since they are treated as strings

In practice, Big SQL uses the Hive catalog. Existing Hive tables can be directly queried by Big SQL and other InfoSphere BigInsights tools. However, Big SQL supports a richer set of SQL functions than Hive supports. For example, Big SQL supports subqueries by using `In` and `Not in` statements, `Having` statements, `With` statements, advanced aggregate functions, and common table expressions. Big SQL provides better performance for simple queries by avoiding the required 15-second latency for Hive query startup time for triggering a Hadoop job.

Cloudera Impala

Impala is an open source massively parallel processing SQL query engine that runs natively in Apache Hadoop. Built by Cloudera, Impala does not use Hadoop to run the queries. Instead, it relies on its own set of daemons, which are installed alongside the data nodes and are tuned to optimize the local processing to avoid bottlenecks. Impala is part of the Hadoop ecosystem and shares infrastructure (for example, metadata, Apache Hive, and Apache Pig). Therefore, by using Impala, the user can query data, regardless of whether it is stored in HDFS or Apache HBase. It

uses the same metadata, SQL syntax (Hive SQL), that Apache Hive uses. Compared to Impala, the Big SQL engine supports much richer SQL dialect and functions. One of the main limitations of Impala limitation is that it relies on in-memory `join` implementation. Therefore, queries can fail if the joined tables can't fit into memory

Pivotal HAWQ

HAdoop With Query (HAWQ) is a closed-source project that is offered by EMC Pivotal as a massively parallel processing database. HAWQ is basically the Greenplum database that is extended to store data on HDFS. HAWQ relies on both the Postgres database and the HDFS storage as its backend storage mechanism. Because it uses Postgres, HAWQ can support the full SQL syntax. The SQL access for the HDFS data is managed by using external tables.

The key architecture of the HAWQ engine includes these components:

- HAWQ master: Responsible for accepting the connections from the clients and managing the system tables that contain metadata information. HAWQ has its own proprietary catalog and does not use HIVE catalog. The master node is also responsible for parsing and optimizing the queries and generating the query execution plan.
- HAWQ segments: Represent the processing units because they are responsible for running the local database operations on their own data sets.
- HAWQ storage nodes: Store all the user data. HAWQ relies on a proprietary file format for storing the HDFS data.
- HAWQ interconnect: Responsible for managing the inter-process communication between the segments during the query execution.

BIG SQL offers smoother integration with the different components of Hadoop ecosystem. However, in addition to its full SQL support, HAWQ can provide strong performance that is based on its heritage as a massively parallel processing engine.

Conclusion

In the last decade, the Hadoop framework has emerged as a popular mechanism to harness the power of large clusters of computers. It enables programmers to think in a data-centric fashion and to focus on applying transformations to sets of data records because the details of distributed execution and fault tolerance are transparently managed by the MapReduce framework. Recently, new domain-specific systems were introduced. These systems seem to be forming a new generation of big data systems. MapReduce is unlikely to be completely replaced by these systems. It is more likely that they will coexist and complement each other in different scenarios.

SQL is the most widely used language to access, analyze, and manipulate structured data. The need for SQL support on the Hadoop environment for processing large structured data is growing rapidly. This article focuses on the SQL-on-Hadoop wave of systems that is gaining a momentum because of the increasing demand to run interactive queries on top of the Hadoop environment for processing massive data sets. The article includes an overview of IBM Big SQL and introduces

three other key systems: Hive, Impala, and HAWQ. However, the spectrum of this wave includes other systems such as Microsoft Polybase, Google BigQuery, Teradata (SQL-H), Hadapt, Apache Drill, Facebook Presto, and SAP HANA. Use the resources included in this article to do your own research on SQL-on-Hadoop systems.

Resources

Learn

- Learn more about InfoSphere BigInsights and refer to the product documentation for details.
- Read "Understanding InfoSphere BigInsights" (Cynthia Saracco, developerWorks, October 2011) to learn more about the product's architecture and underlying technologies.
- Watch BigInsights experts discuss the technology, give demos, and answer common questions on the IBM big data channel on YouTube.
- Get a technical introduction to Big SQL on Slideshare.
- Read "What's the big deal about Big SQL?" (developerWorks, June 2013) to learn about IBM's SQL interface to its Hadoop-based platform, InfoSphere BigInsights. Big SQL is designed to provide SQL developers with an easy solution for querying data that is managed by Hadoop.
- Dig deeper in the "Developing Big SQL queries to analyze big data" tutorial in the InfoSphere BigInsights tutorial collection (PDF).
- Find out more about Apache Hadoop and learn about Apache Hive Project.
- Explore "The family of MapReduce and large-scale data processing systems" (Sherif Sakr, Anna Liu, Ayman G. Fayoumi, ACM Digital Library, 2013).
- Get "A comparison of approaches to large-scale data analysis" (Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, Michael Stonebraker, SIGMOD Conference 2009).

Get products and technologies

- Download InfoSphere BigInsights Quick Start Edition, available as a native software installation or as a VMware image.

Discuss

- Discuss Big SQL and connect with other BigInsights users through the BigInsights forum.

About the author

Sherif Sakr

Dr. Sherif Sakr is a senior research scientist in the Software Systems Group at National ICT Australia (NICTA), Sydney, Australia. He is also a conjoint lecturer in the School of Computer Science and Engineering at University of New South Wales. He received his doctorate in computer science from Konstanz University, Germany, in 2007. His bachelor's and master's degrees in computer science are from Cairo University, Egypt. In 2011, Dr. Sakr held a visiting research scientist position in the eXtreme Computing Group (XCG) at Microsoft Research, in Redmond, Wash. In 2012, he held a research MTS position in Alcatel-Lucent Bell Labs. Sherif is a Cloudera certified developer for Apache Hadoop and Cloudera certified Specialist for HBase.