International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Impala: Open Source, Native Analytic Database for Apache Hadoop - A Review Paper

Prof. Pramod Patil[1], Amit Patange[2]

[1]Department of Computer Engineering DYPIET Pimpri, SavitriBai Phule Pune University, India
[2]Department of Computer Engineering DYPIET Pimpri, SavitriBai Phule Pune University, India

Abstract: ***Cloudera impala is today's modern, open source massively parallel processing database on top hadoop data processing environment. Impala database provides high performance queries, low-latency and high concurrency for business intelligence application. Impala runs and gives us output in real-time. Impala sets new benchmarks for hadoop databases. As comparative to Apache pig scripts and hive queries impala shows a better performance in all the aspects.***

Keywords: Cloudera Impala, MPP database, Data Analytics, HDFS, HBase

1. Introduction

Impala is one of the open source massively parallel processing (MPP) analytic database for Apache Hadoop[10] in the Big Data stream. Cloudera launches impala in the mid of October 2012[15]. Impala is integrated with native Hadoop security and Kerberos for authentication, and via the Sentry module, you can ensure that the right users and applications are authorized for the right data. Impala has set a new standard benchmarks against which alternatives measure themselves, based on a proliferation of new benchmark testing. Impala has been adopted by multiple vendors as their solution for letting customers do exploratory analysis on Big Data, natively and in place. Impala raises the bar for query performance while retaining a familiar user experience. A massively parallel processing SQL engine for interactive analytics and business intelligence. Its highly optimized architecture makes it ideally suited for traditional BI-style queries with joins, aggregations, and sub queries. It can query Hadoop data files from a variety of sources, including those produced by MapReduce jobs or loaded into Hive tables. Unlike other systems (often forks of Postgres), Impala is a brand-new engine, written from the ground up in C++ and Java. It maintains Hadoop's flexibility by utilizing standard components (HDFS, HBase, Metastore, YARN, Sentry) and is able to read the majority of the widely-used file formats (e.g. Parquet, Avro, RCFile). This paper discusses the services Impala provides to the user and then presents an overview of its architecture and main components. The highest performance that is achievable today requires using HDFS as the underlying storage manager, and therefore that is the focus on this paper; when there are notable differences in terms of how certain technical aspects are handled in conjunction with HBase.

2. Literature Survey

Large-scale analytical data processing has become widespread in web companies and across industries, not least due to low-cost storage that enabled collecting vast amounts of business-critical data. Performing interactive data analysis at scale demands a high degree of parallelism. For example, reading one terabyte of compressed data in one second using today's commodity disks would require tens of thousands of disks. Similarly, CPU-intensive queries may need to run on thousands of cores to complete within seconds. At Google, Dremel is a scalable [16], interactive ad-hoc query system for analysis of read-only nested data. By combining multi-level execution trees and columnar data layout, it is capable of running aggregation queries over trillion-row tables in seconds. The MapReduce (MR) framework was designed to address the challenges of large-scale computing in the context of long-running batch jobs. Like MR, Dremel provides fault tolerant execution, a flexible data model, and in situ data processing capabilities. Dremel, a distributed system for interactive analysis of large datasets. Dremel is a custom, scalable data management solution built from simpler components. As like Dremel Google also introduced 'Big Tables' as it is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Google have designed, implemented, and deployed a distributed storage system for managing structured data at Google called Bigtable [22]. Bigtable is designed to reliably scale to petabytes of data and thousands of machines. Bigtable has achieved several goals: wide applicability, scalability, high performance, and high availability. After this research Cloudera Impala launches high-performance, low-latency SQL queries on data stored in popular Apache Hadoop file formats. The fast response for queries enables interactive exploration and fine-tuning of analytic queries, rather than long batch jobs traditionally associated with SQL-on-Hadoop technologies. The latest release of Cloudera impala is Impala 2.1.0 and it is launched in Dec-2014.

3. Limitations of Existing Systems

Impala doesn't have to translate a SQL query into another processing framework, like the map/shuffle/reduce operations on which Hive and Apache pig depends today. As a result, Impala doesn't suffer the latencies that those operations impose. An Impala query begins to deliver results right away. Because it's designed from the ground up for SQL query execution, not as a general-purpose distributed processing system like MapReduce, it's able to deliver much better performance for that particular workload [17]. Impala came into picture because Hive is the wrong architecture for real-time distributed SQL processing. The landscape of parallel

SQL databases is densely populated. No traditional relational vendor — IBM, Oracle, Microsoft, ParAccel, Greenplum, Teradata, Netezza, Vertica — uses anything like MapReduce. Facebook built Hive on MapReduce early because it was the shortest path to SQL on Hadoop. It's certainly possible to improve Hive, and to tune MapReduce, to speed things up and to reduce latencies. By design, though, Hive will always impose overhead and incur performance penalties that successor systems, built as native distributed SQL engines, avoid. Twitter, Cloudera and Criteo collaborate on Parquet, a columnar format that lets Impala run analytic database workloads much faster. Contributors are working on integrating Parquet with Cascading, Pig and even Hive. User- and role-based authentication and access control are crucial for database applications and this is what impala supports. Impala performs in-memory query processing while Hive does not. Hive use MapReduce to process queries, while Impala uses its own processing engine. Hive can be extended using User Defined Functions (UDF) or writing a custom Serializer/Deserializer.

4. Cloudera Impala Architecture

Hadoop is a batch oriented solution that has a lack of support for ad-hoc, real-time queries. Many of the players in Big Data have realized the need for fast, interactive queries besides the traditional Hadoop approach [18]. Cloudera, one the key solution vendors in Big Data/Hadoop domain has just recently launched Cloudera Impala that addresses this gap. The work was inspired by Google Dremel paper which is also the basis for Google BigQuery. Cloudera Impala provides a HiveQL-like query language for wide variety of SELECT statements with WHERE, GROUP BY, HAVING clauses, with ORDER BY – though currently LIMIT is mandatory with ORDER BY -, joins (LEFT, RIGTH, FULL, OUTER, INNER), UNION ALL, external tables, etc. It also supports arithmetic and logical operators and Hive built-in functions such as COUNT, SUM, LIKE, IN or BETWEEN. It can access data stored on HDFS but it does not use MapReduce, instead it is based on its own distributed query engine.

Cloudera Impala has 3 key components: impalad, impala-state-store and impala-shell.

1. **Impala shell** is essentially a short shell script which starts the impala-shell.py python program to run the queries.
2. **Impalad** is running on each Hadoop data node and it plans and executes the queries sent from impala-shell.
3. **Impala-state-store** stores information (location and status) about all the running impalad instances.

To avoid latency, Impala circumvents MapReduce to directly access the data through a specialized distributed query engine [19] that is very similar to those found in commercial parallel RDBMSs. The result is order-of-magnitude faster performance than Hive, depending on the type of query and configuration. Note that this performance improvement has been confirmed by several large companies that have tested Impala on real-world workloads for several months now. Here is the core architecture of Cloudera impala as follows,

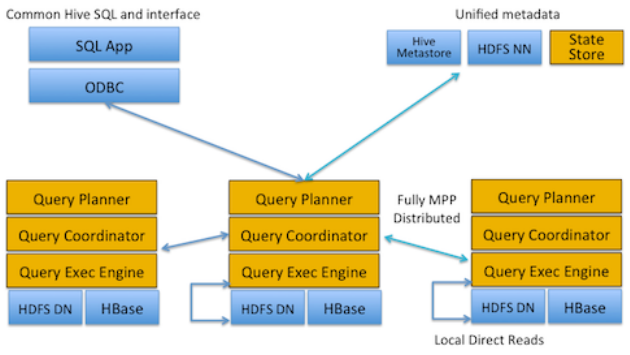

Figure 1: Core Architecture of Cloudera Impala

HBase [20] is a distributed column-oriented database built on top of the Hadoop file system. It is an open-source project and is horizontally scalable. HBase is a data model that is similar to Google's big table designed to provide quick random access to huge amounts of structured data. It leverages the fault tolerance provided by the Hadoop File System (HDFS) [9]. It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System. One can store the data in HDFS either directly or through HBase. Data consumer reads/accesses the data in HDFS randomly using HBase. HBase sits on top of the Hadoop File System and provides read and write access. All the metadata for Hive tables and partitions are accessed through the Hive Metastore. Metadata is persisted using JPOX ORM solution (Data Nucleus) so any database that is supported by it can be used by Hive. Most of the commercial relational databases and many open source databases are supported. See the list of supported databases in section below. The precise amount of performance improvement is highly dependent on a number of factors:

Hardware configuration: Impala is generally able to take full advantage of hardware resources and specifically generates less CPU load than Hive, which often translates into higher observed aggregate I/O bandwidth than with Hive. Impala of course cannot go faster than the hardware permits, so any hardware bottlenecks will limit the observed speedup. For purely I/O bound queries, we typically see performance gains in the range of 3-4x. Complexity of the query: Queries that require multiple MapReduce phases in Hive or require reduce-side joins will see a higher speedup than, say, simple single-table aggregation queries. For queries with at least one join, we have seemed performance gains of 7-45X. Availability of main memory as a cache for table data: If the data accessed through the query comes out of the cache, the speedup will be more dramatic thanks to Impala's superior efficiency. In those scenarios, we have seen speedups of 20x-90x over Hive even on simple aggregation queries. We ran some tests on top of hadoop databases and we got following results as follows.

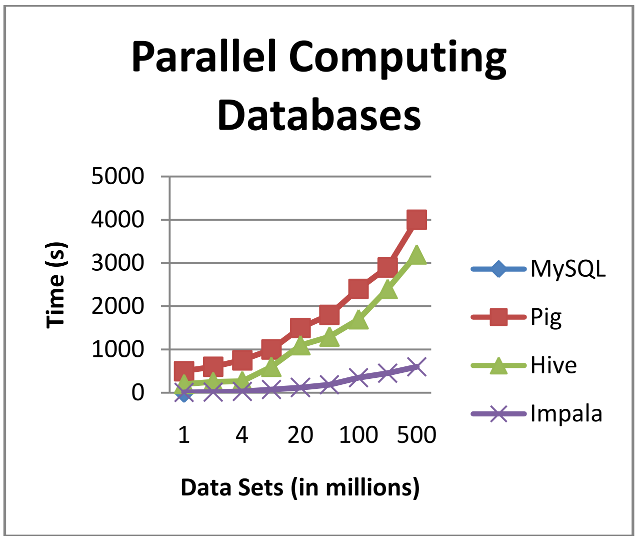

Figure 2: Parallel Computing database comparison

Conclusion

In this paper we presented Cloudera Impala, an open source SQL engine that was designed to bring parallel DBMS technology to the Hadoop environment. Our performance results showed that despite Hadoop's origin as a batch processing environment, it is possible to build an analytic DBMS on top of it that performs just as well or better those current commercial solutions, but at the same time retains the flexibility and cost-effectiveness of Hadoop. In its present state, Impala can already replace traditional, monolithic analytic RDBMSs for many workloads. There are more and more efforts in the Big Data world to support ad-hoc, fast queries and real-time data processing for large datasets. Cloudera Impala is certainly an exciting solution that is utilizing the same concept as Google BigQuery but promises to support wider range of input formats and by making it available as an open source technology it can attract external developers to improve the software and take it to the next stage.

Acknowledgement

I would like to take this opportunity to acknowledge the contribution of certain people without which it would not have been possible to complete this paper work. I would like to express my special appreciation and thanks to my advisor Professor Pramod Patil, you have been an overwhelming mentor for me.

References

[1] Bosworth, Gray, Chaudhuri, Reichart, Pellow, Venkatrao, "Data cube: a relational operator generalizing group by, cross-tab and sub-totals, " Proc. 12th Int'l Conf. Data Eng. (ICDE), 1996.
[2] Deshpande, S. and R. Agarwal, Gupta, Naughton J., Sarawagi, Ramakrishnan, "On the computation of multidimensional aggregates," Proc.22nd Int'l Conf. Very Large Data Bases (VLDB), 1996.
[3] M. Deshpande, F. Naughton, Zhao."An array based algorithm for simultaneous multidimensional aggregates". In SIGMOD'97.
[4] Srivastava D., Ross, "Fast computation of sparse data cubes," Proc. 23rd Int'l Conf. Very Large Data Bases (VLDB), 1997.
[5] Han, Xin D., X. Li, and W. B. Wah Starcubing: Computing iceberg cubes by top-down and bottom-up integration. In VLDB'03.
[6] S.Wagner, R.T. Ng and Yin Y., "Iceberg-cube computation with PC clusters," Proc. ACM SIGMOD Int'l Conf. Management of Data, 2001.
[7] Yury K. and Sergey K., "Applying map-reduce paradigm for parallel closed cube computation," Proc. First Int'l Conf. Advances in Databases, Knowledge, and Data Applications (DBKDA), 2009.
[8] S. Ghemawat, Dean J. MapReduce: Simplified Data Processing on Large Clusters. OSDI, 2004.
[9] Abouzeid A. et al., "HadoopDB: an architectural hybrid of mapreduce and DBMS technologies for analytical workloads," Proc. VLDB Endowment, vol. 2, pp. 922-933, 2009.
[10] Murthy A.C. and Shvachko K.V., "Scaling hadoop to 4000 nodes at Yahoo!," Yahoo! Developer Network Blog, 2008.
[11] Yu Cong, Arnab Nandi, Bohannon Philip, and Ramakrishnan Raghu "Data cube materialization and mining over map-reduce " IEEE transaction on Knowledge and Data Engineering, vol. 24, no. 10, Oct 2012.
[12] C. Yu, Bohannon P., Nandi A. and Rama krishnan R., "Distributed cube materialization on holistic measures," Proc. IEEE 27th Int'l Conf. Data Eng. (ICDE), 2011.
[13] Chen Y., Dehne F. K. H. A., Eavis T., and Rau Chaplin A. PnP: Sequential, external memory and parallel iceberg cube computation. Distributed and Parallel Databases, 2008.
[14] Sergey K. and Yury K. Applying Map Reduce Paradigm for Parallel Closed Cube Computation. DBKDA, 2009.
[15] http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/introduction.html
[16] http://theprofessionalspoint.blogspot.in/2012/12/google-dremel-vs-apache-hadoop-big-data.html
[17] http://vision.cloudera.com/impala-v-hive/
[18] https://wishkane.wordpress.com/2013/07/06/cloudera-impala-fast-interactive-queries-with-hadoop/
[19] http://blog.cloudera.com/blog/2012/10/cloudera-impala-real-time-queries-in-apache-hadoop-for-real/
[20] http://www.tutorialspoint.com/hbase/hbase_overview.htm
[21] https://wishkane.wordpress.com/2013/07/06/cloudera-impala-fast-interactive-queries-with-hadoop/
[22] http://research.google.com/archive/bigtable.html

Author Profile

Prof. Pramod D. Patil obtained his Bachelor's degree in Computer Science and Engineering from Swami Ramanand Tirth Marathwada University, India. Then he obtained his Master's degree in Computer Engineering and pursuing PhD in Computer Engineering majoring in Mining Data Streams both from Pune University, INDIA. Currently, he is a Research Scholar in Department of Computer Engineering at COEP, Pune University, INDIA. His specializations include Database Management System, Data Mining, and Web

Mining. His current research interests are Mining Data Streams.

Mr. Amit Patange obtained his Bachelor’s degree in Computer Science from University of pune, India. Now pursuing Master’s degree in Computer Engineering University of pune, INDIA. His dissertations work on Data Mining.