***Review : SQL-on-Hadoop: Full Circle Back to Shared-Nothing Database Architectures***

***Problem:***

* Use of MapReduce framework to processes SQL statements has increased rapidly in Analytical Systems.
* This paper compares the performance of - Hive and Impala - two SQL-on-Hadoop systems
* It studies the role of the file formats in I/O effectiveness and columnar stores in performance of these systems.

***Why the problem hard***

* Setting up experiments and benchmarking the two systems in the exact same conditions is non-trivial.

***Why is it important***

* Many businesses today use Hadoop as the repository to store data coming from various sources and to perform analytics
* Various frameworks are used to run analytics
  + Text analytics on unstructured data
  + Log analysis on semi-structured data
  + SQL-like processing over semi-structure and structured data.

***Main idea:***

There are 2 broad categories of SQL on Hadoop Systems

- Native Hadoop-based

* Hive:
  + first SQL-like system over Hadoop
  + uses another framework such as MapReduce or Tez to process SQL-like queries.
* Shark: similar to Hive – just uses a different framework Spark
* Impala: runs query on each node using its own long-running deamon processes.

- Database Hadoop-hybrids

* uses relational database to execute query

This paper focuses on the first category and compares Hive with Impala

***Background :***

***Impala:***

* Supports Sql processing over Hadoop
* Has a highly efficient I/O layer
  + Impala’s I/O subsystem provides much faster ingestion rates
* Exploits the streaming SIMD
* Streams the data between stages of the query computation, resulting in significant performance improvements
* Cannot recover from mid-query failures yet, as it needs to re-run the whole query in case of a failure

***Hive:***

* Built on Hadoop
  + all advantages of hadoop - such as fault tolerance, high availability and scaling to 1000s of nodes
* Uses ORC - which is the columnar style file format for HDFS
  + ORC tends to pre fetch unnecessary data especially when a table contains a large number of column
* Hive on MapReduce is also impacted by the startup and scheduling overheads of the MapReduce framework, and pays the cost of writing intermediate results into HDFS.
* Hive on Tez eliminates the startup and materialization overheads of MapReduce, but has the Java deserialization overheads during scan operations and thus it is still slower than Impala

***Conclusion:***

* Through various experiments and tests the paper proves that the performance of Impala provides a significant benefit over Hive when the data fits in memory
  + This is mostly to very efficient I/O and pipelined query execution - which close to shared nothing database.
* Hive and Tez are CPU bound during scan operations and impact performance.

***Next Steps:***

* See if the Scan operation of Tez / Hive be optimized such that they are not CPU bound
* Improve the I/O functionality of Hive/Tez