1) A. THUSOO, J. SEN SARMA, N. JAIN, Z. SHAO, P. CHAKKA, N. ZHANG, S. ANTONY, H. LIU, AND R. MURTHY. HIVE – A PETABYTE SCALE DATA WAREHOUSE USING HADOOP, 2010.

2) A. PAVLO, E. PAULSON, A. RASIN, D. J. ABADI, D.J. DEWITT, S. MADDEN AND M. STONEBRAKER. A COMPARISON OF APPROACHES TO LARGE-SCALE DATA ANALYSIS.

RAHUL NARAYAN

7TH MAY 2014

MAIN IDEA: HIVE – A PETABYTE SCALE DATA WAREHOUSE USING HADOOP

- Presenting one among several options for collecting and analyzing large sets of data (which increase exponentially in size over time) required for business intelligence/activities due to the prohibitive costs of using traditional data warehousing to store this amount of information.
- Presenting and describing Hadoop: a popular open-source map-reduce implementation along with Hive: an open-source data warehousing solution built on top of Hadoop as an option to process and store growing data needs experienced at Facebook.
- Providing technical details on map-reduce implementations to those that may not be familiar with it by presenting map-reduce in a more "SQL friendly" light.

IMPLEMENTATION OF HIVE

- Comparing running times (and speculated running times) of queries written in SQL for business requirements and comparing those to Hive’s map-reduce.
- Trying to see whether the above will be scalable.
- Presenting technical details of Hive including native data structures, types, query languages (including sample syntax), query compiler, the metastore (essentially the Hive syscatalog), file formats, and current usage at Facebook.
- Examples comparing sample SQL code required to perform a business operation in a traditional DBMS environment to map-reduction programs written to perform the same operations under Hive.

ANALYSIS

- There is a “HiveQL” language which is meant to mimic mainstream SQL which will help in using Hive (Example: primitive types supported by SQL are supported by HiveQL). However, there are nuances to this which are motivated specifically by the experiences gained at Facebook.
- In order to reduce search time, there are certain options available within HiveQL like creating partitions. This will help the Hive compiler prune directories that need to be scanned to execute a business query.
- Map-reduce programs must be written in order to glean data from this system which can be very complicated. Any minor changes may render previously written map-reduce programs useless when used to gather information from the system.
- In order to address scalability, Hive ensures that no Metastore (Hive syscatalog) calls are made from the mappers or reducers of a job.

COMPARISON TO PAPER 2

- The second paper takes a generally objective view at data warehousing solutions by offering a more thorough analysis and comparison between parallel DBMS and MR for both small data sets and large data sets. The first paper focuses more on the technical aspects of MR.
- While addressing MR, the second paper also alludes to the overhead costs (in a fair bit of detail) not only of MR, but also of parallel DBMS which the first doesn’t seem to do.
- The first paper does a more through job explaining why it may sometimes take a while for MR to run certain queries by addressing the metastore associated with Hive.
- The first paper presents MR implementations in heavy technical detail in the light of an actual business case while the second looks at a more generic perspective on MR.

ADVANTAGES/DISADVANTAGES OF MAIN IDEA

- Advantages:
  - For data loading, Hadoop outperforms parallel DBMS since each node is copying data files from the local disk into the local HDFS and distributing replicas to other nodes in the cluster.
  - Easier to install and configure properly compared to parallel DBMS.
  - If data is only going to be loaded once for certain analysis types of tasks, it may not be worth it to pay the extra indexing and re-org costs of a DBMS.
  - Good for aggregation.
- Disadvantages:
  - Hadoop has significant startup costs. This becomes an issue when we're dealing with small amounts of data. Doesn't make sense to pay all that overhead.
  - If we want to perform something like an analytical selection task (get PageRank higher than a certain value) parallel DBMS is much better since data is probably indexed on the field we're searching.
  - More difficult to do joins on MR because there is no inherent ability to join 2 or more disparate datasets.
  - Parallel DBMS are always "warm" and ready to accept queries while Hadoop may not be.
  - Maintenance of MR programs is very difficult.