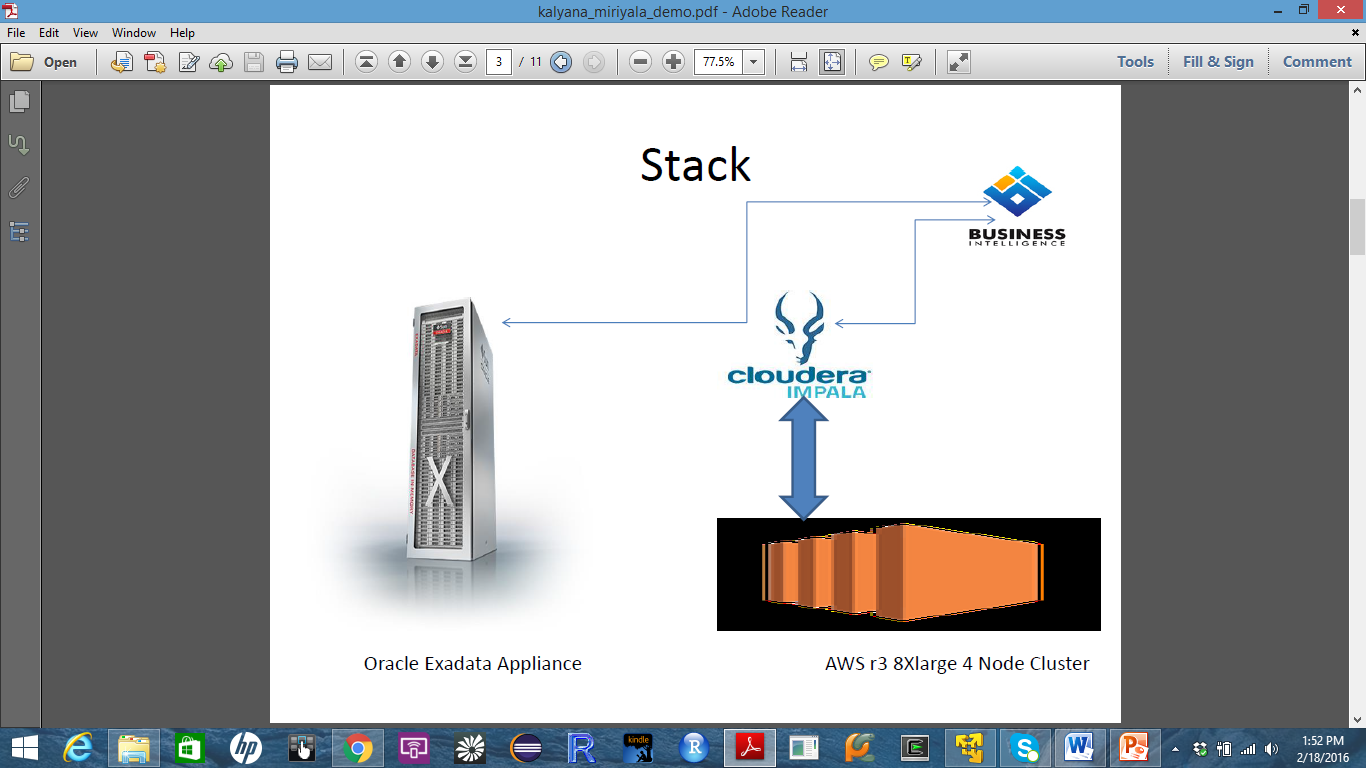
# Hadoop Data Warehouse

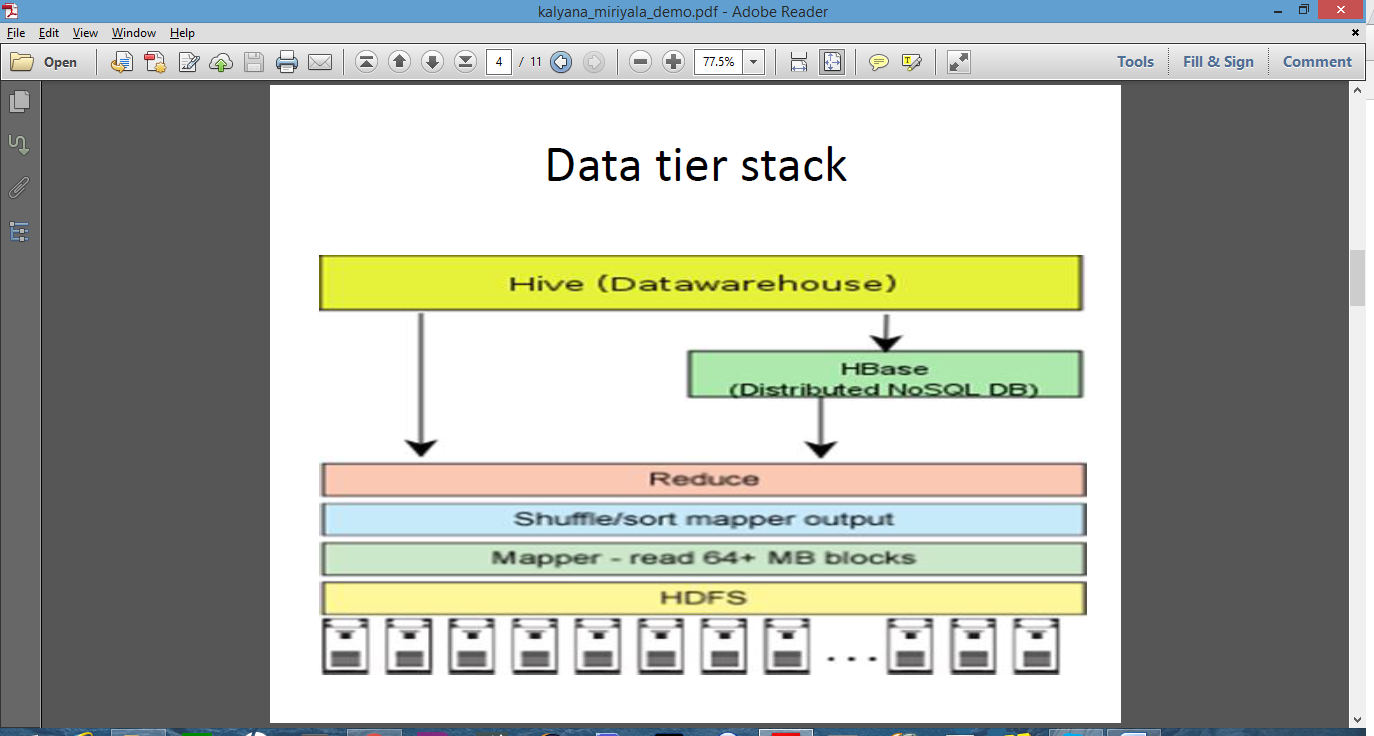
## Impala and HBase stack for slowly changing dimensions in Hive DW

Hadoop offers a scalable platform for storage, search and analytics at attractive price point. It offers an alternative to proprietary data warehouses at potentially lower price point with richer analytical capabilities. Hadoop HDFS and Hive are typically append-only filesystems, with no ability to modify the data after the writes. I have tested a stack that will accommodate slowly changing dimensions with ease when updates to those values come through from the source system; this will facilitate moving some of the functionality from expensive proprietary data warehouses to Hadoop stack.

The Hadoop stack consists of Cloudera Impala Query engine running on AWS Cluster for interactive analytics.



Underneath the hood Impala relies on Hive Data Warehouse for cataloging database objects. The data tier consists of HDFS Parquet storage for Fact tables and HBase for Dimension tables.



Apache HBase, a NoSQL database sits over HDFS but allows updates and deletes to individual rows of data rather than restricting to just append/inserts. HBase is a Key/Value-columnar store where individual rows have a key, and one or more column families made up of one or more columns. While defining a HBase table you only define column families, and data load itself creates the columns within them. HBase is primarily designed for low-latency inserts and row retrievals, hence well suited for data warehouse dimension tables.

Parquet storage is built for interoperability, space and Query efficiency, hence well suited for hosting Fact tables. A subsequent change to existing data necessitates dropping the partition and reloading the new data.

For testing purposes I had a need to ingest large volumes of data, TPC-DS toolkit provided by Cloudera was used to build the Sales Data Warehouse. For dimension tables I have used Hive integration with HBase to copy data. Bulk loading data into these Hive-on-HBase tables is then just a matter of loading the source data into regular Hive table, and then running INSERT INTO TABLE…SELECT to copy regular Hive rows into HBase tables via their Hive metadata overlays.

The graph below reflects the storage savings for the fact table after copying text data in to parquet format. The parquet format resulted in 70% storage savings.



The graph below reflects the performance of Impala queries. I ran 20 Queries to assess the performance of the stack. The queries were categorized as Interactive and Reporting. Interactive queries surveyed data on smaller data sets vs reporting queries which had much wider coverage. The queries were run on 3 sets of storage formats. The first format was all data in text (facts and dimensions) for baselining. The second format was all data in parquet (facts and dimensions). The third format was target stack with facts in parquet and dimensions in Hbase. For this use case the reporting queries ran on average 85 sec with all data in text vs 34 secs for all data in parquet vs 37 secs for target stack with fact in parquet and dimensions on HBase.



This test is not a fill-blown benchmarking study, since there are many important parameters to consider, e.g. data generated for this test is synthetic via a tool kit, and does not reflect the volumes and data patterns of a data warehouse, the specs of the nodes to just name a few. However, just taking the technologies out of the box, playing around with them and stress-testing them in a simple yet reasonable test environment provides valuable insights on the capabilities of Impala, Hive, Parquet and Hbase on Hadoop and potential benefits it can bring to an enterprise for data warehousing and analytics.