\section{Implementation}

Multiple vendors provide platforms and infrastructure for data warehousing technologies,

this project was implemented using Amazon Web Services (AWS).

AWS is a set of cloud computing services provided by Amazon.com\footnote{https://aws.amazon.com/}

that are accessible over the internet.

AWS provides multiple services for many different applications.

Elastic MapReduce (EMR) was used for the implementation of Hadoop in this project.

\subsection{AWS Elastic MapReduce}

AWS EMR is a pre-configured compute cluster for Big Data.

The cluster can be provisioned and terminated on demand as needed.

It comes with a configurable set of the Hadoop ecosystem elements pre-installed and ready to use.

Fig. \ref{EMR} shows the ecosystem of EMR ecosystem.

The EMR cluster used in this study was provisioned with three

m5.xlarge\footnote{https://aws.amazon.com/ec2/instance-types/m5/}

elastic compute instances using version 5.27.0 of the EMR

software\footnote{emr-5.27.0 contains Amazon Hadoop 2.8.5, Hive 2.3.5, and Hue 4.4.0. See https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-release-5x.html}.

\subsection{Apache Hadoop}

Hadoop is an open source software framework for the distributed storage and

distributed processing of a very large datasets.

The core of Apache Hadoop consists of a storage part: Hadoop Distributed File System

(HDFS) and a processing part MapReduce.

\subsubsection{Hadoop Distributed File System}

Unlike traditional file system, HDFS is a distributed file system designed to run on commodity hardware.

HDFS the architecture consists of a master (NameNode) and at least one slave (DataNodes)\cite{HDFSarchitecture}.

The NameNode is the controller which consist storing data and metadata of the data no the DataNodes.

The metadata includes a Namespace lookup table used to locate each file from the DataNodes.

The Fig. \ref{HDFS} shows the Architecture of HDFS \cite{HDFS}.

\subsubsection{Hadoop MapReduce}

Hadoop MapReduce is a programming model for large scale data processing.

With the high demand of computing power,

computer architecture has evolved over the years from serial computing to parallel computing

where tasks are distributed and executed simultaneously across multiple processors within the same computer.

With MapReduce,

instead of using one server instance with multiple processors,

multiple servers with multiple processors are used for computation.

This is called distributed computing system.

The parallelism in Hadoop is categorized into data-parallelism and task-parallelism.

Data-parallelism is focused on processing data across multiple processors whereas

task-parallelism is focused on distributing execution threads across multiple servers \cite{Parallelism}.

Utilizing one or the other is dependent upon whether data is being provided into

the system or being utilized for events such as machine learning.

This is what enables management and processing of "Big Data" (multi-terabyte datasets)

in parallel on large clusters (thousands of nodes) of commodity hardware in a reliable,

fault-tolerant manner.

A MapReduce framework consists of two types of workers: mappers and reducers.

MapReduce uses the divide and conquer technique where the input is divided into a set

of small tasks and each task is identified by a key-value pair \cite{Divide-and-Conquer}.

The key serves as a task ID and the value as the task output.

Each task is then processed and executed by the mapper and

the outputs are processed and merged by the reducer \cite{MapReduce}.

\begin{figure}

\centering

\includegraphics[width=2.5in]{EMR\_Ecosystem.png}

\caption{Elastic MapReduce Ecosystem}

\label{EMR}

\end{figure}

\subsection{AWS Simple Storage Service}

As the name implies, the S3 is a storage system.

It is used to store and retrieve any amount of data any time, from anywhere on the web.

S3 is reliable, fast, and inexpensive.

\subsection{Apache Hive}

Apache Hive is a data warehouse application built on top of Hadoop. We used Hive to structure, organize, model, and query our data using the Hive Query Language, HQL. We structured files from within the Hadoop Distributed File System (HDFS) and loaded them into tables within Hive. Hive operates by using the HDFS storage to generate tables. However, when Hive queries are executed, the data is retrieved from HDFS via the compiler, which uses an execution engine and metastore to return results. See: <http://www.guardxinc.com/blog/2017/04/10/hive-queries-whats-really-going-on-in-there/>

\subsection{Cloudera Hue}

Cloudera Hue is an open source web application that served as user interface for Hadoop components.

Hue provides access to Hadoop from within a browser, allowing users to interact with the Hadoop ecosystem applications.

Hue is an alternative to accessing the Hadoop ecosystem applications from the command line interface.

Hue was used to interact with the EMR cluster and run Hive scripts.

\begin{figure}

\centering

\includegraphics[width=2.5in]{HDFS\_Arch.png}

\caption{HDFS Architecture \cite{HDFS}}

\label{HDFS}

\end{figure}

\section{Study Design}

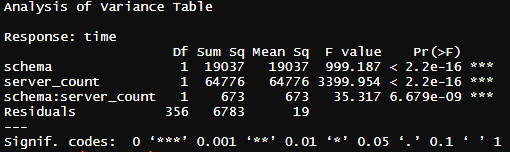
During performance testing, we gathered all fields from all tables, joining all tables in the schema together. Joining all tables enabled capturing the entirety of the database, under each schema, to produce the same results. These queries were limited to 75,000 returned records to enable gathering more runtime samples. This number was decided to be reasonable based on the difference in performance time. As noted in the analysis section below, the performance time between queries was significant at 75,000 records; limited a higher volume of returned records would have increased the significance. Therefore, 75,000 was determined reasonable for analysis.

\section{Results}

As discussed in the Study Design section, a two-way Analysis of Variance (ANOVA) test was performed. The conclusion of this analysis is that – while holding cluster size constant – schema design is significant on performance across both server sizes (p-value < 0.0001, F-statistic 999.187). Furthermore, while holding schema design constant, performance speed difference was also significant (p-value < 0.0001, F-statistic 3399.954) between the two cluster sizes with the five-server cluster outperforming the three-server cluster. Finally, we also identified through the two-way ANOVA that the interaction between schema design and server count provide significant results, indicating that there is a difference in the proportions of performance speed between the two schemas when considering count of servers in a cluster.

In analyzing the exploratory graphical analysis of the clusters, the interaction between schema and cluster size is such that as cluster size increases, the normalized schema approaches a similar level of performance. However, testing indicates there is still a statistically significant difference between performance at the five-server cluster level. Further testing could identify that – while holding database size constant – at some server level, the normalized schema may outperform the de-normalized schema. However, this is based on linear approximation that may become non-linear with more samples and clusters. Regardless, under both cluster sizes, the de-normalized schema outperformed the normalized schema by a level of statistical significance.

Maybe a table comparing read times for the different options.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Schema** | **Servers** | **Mean** | **Median** | **Std. Dev.** | **Variance** | **Min** | **Max** |
| Normalized | Three Servers | 121.05 | 120.00 | 4.09 | 16.76 | 114.91 | 131.56 |
| Denormalized | Three Servers | 103.77 | 104.31 | 2.97 | 8.81 | 97.74 | 118.64 |
| Normalized | Five Servers | 91.49 | 92.16 | 3.98 | 15.87 | 81.76 | 99.16 |
| Denormalized | Five Servers | 79.68 | 77.26 | 5.82 | 33.92 | 72.20 | 92.48 |