Assignment-2 Total = 50pts.

Learning Outcomes

Understand the concepts of parallel and distributed databases including the map-reduce parallelism framework, NoSQL, Spark, data replication and distributed query processing.

1. Download Apache Hadoop from

https://hadoop.apache.org/ and test it [5pts + 5pts]. Take screen shots for your test and write how you tested the correct installation. Here, it is not necessary for you to demonstrate multinode cluster.

You may take the help from the book by Hadoop: Definitive Guide: Tom White and the links in the Resource1 in the folder, Week1.

This is the tut I followed (<https://www.digitalocean.com/community/tutorials/how-to-install-hadoop-in-stand-alone-mode-on-ubuntu-16-04>)

Changed block size



Checked namenodes



2. Download Apache Spark from

https://spark.apache.org/and test it. [5pts + 5pts]

Tutorial Links.

[Big Data Processing with Apache Spark - Part 2: Spark SQL (infoq.com)](https://www.infoq.com/articles/apache-spark-sql/)

<https://mapr.com/blog/using-apache-spark-dataframes-processing-tabular-data/>

You may test out Apache Spark with excel or a text file. You may create your own data file. Take screen shots for your test and write how you tested the correct installation.

I installed pyspark an created a dataframe following the first tutorial link. This can be seen in the screenshot below

A screenshot of a computer program

Description automatically generated with medium confidence

3. What are the differences in the architecture of MapReduce and Apache Spark? [5pts]

Map reduce is a disk based computation framework meaning it writes intermediate data to disk which can be slower due to disk I/O. spark is an in-memory computation framework that stores intermediate data in memory, making it faster.

Map reduce has a more linear dataflow structure that works in two stages, map and reduce. On the other hand spark can perform operations via DAG (directed acyclic graph) which allows more complex computations.

4. What is distributed query processing? Give an example. Does MapReduce and Apache Spark allow distributed processing? [ 5pts + 5pts + 5pts]. Read Chapter 22.1 and Chapter 22.10.

Distributed query processing is a method in which the processing of a database query is shared across multiple machines like gfs. suppose you have a database that's distributed across different geographic locations. When a query happens the database decides which locations to access and how to combine the results to answer the query. MapReduce and Spark do allow for distributed processing. They break down tasks into smaller subtasks that are processed in a distributed manner across nodes in a cluster.

5. Read the paper by The Google Filesystem (See the Reading Folder, file= gfs-sosp2003.pdf).

Write a one page summary for the paper (single spaced 11pts). The audience(reader) of the paper should be a student who has the knowledge of undergraduate database and operating system. Given the page limit is one, you have to pick the main idea behind the paper and give example. [10pts]

You will upload word documents with screenshots testing of Apache Hadoop and Apache Spark. Also, the answers for question (2) and question (3) should be in the document. Upload the dataset that you created. The purpose of this homework is to learn the concepts used MapReduce and Spark.

MY PAPER:

Google File System is a distributed file system created by Google. GFS provides reliable access to data by leveraging a network of inexpensive, low-performance parallel computing components that work in parallel giving Google the most bang for their buck.

The key goals of GFS are:

1. **Performance**: GFS is meant to work in sync with parallel computing, enabling thousands of interconnected machines to work in tandem for optimal read/write speeds.
2. **Scalability**: As the system expands, its performance should not degrade but rather improve or remain consistent. In other words, it should be easy to add more capacity without hindering performance.
3. **Reliability**: Google wants a sturdy system. They achieve this mainly through redundancy.

One strategy GFS uses to achieve these goals is dividing large files into manageable chunks or blocks. Imagine trying to pick out a line from a book that's thousands of pages thick. Flipping through the whole book would take forever. To get around this issue GFS splits up the data into smaller bits, making it way easier to index to the relevant data.

Let's take a deeper dive into GFS's architecture. Picture it this way: GFS is like a librarian (the master) who doesn't read all the books but knows exactly where each one (the chunks) is. Clients ask the librarian for a specific book, and the librarian directs them to the right shelf (the chunk servers). The books themselves are stored in the chunk servers, which are attached to file stores. Each book has a unique ID (chunk handle), and the chunk servers store these books in their own local Linux file systems.

Now, how does a client read a file? Simple. The client sends a request to the master, like asking a librarian for a book by name and page number. The master figures out which book and page (or chunk offset) they need and directs them to the right shelf (chunk server). The client then takes this info to the chunk server, and voila! They've got their data.

In conclusion, GFS's approach to data management is all about getting the best out of parallel computing, ensuring scalability, and maintaining reliability through redundancy. By organizing massive amounts of data into manageable chunks and using a clever system of masters and chunk servers, GFS gives Google a cost-effective way to manage big data.