**Lab 02: Data Storage, MapReduce and Hadoop Pros and Cons**

Steven Simonsen

School of Engineering and Computational Sciences, Merrimack College

DSE6220 Big Data: Hadoop and Spark

Professor Jeremiah Lowhorn

May 19, 2024

**Data Storage, MapReduce and Hadoop Pros and Cons**

The use of Parquet, ORC, or AVRO in a Hadoop environment largely depends on the Big Data platforms utilized. To begin, I will outline the most common use cases, advantages and disadvantages, and examples of Parquet. Most commonly, Parquet is utilized in conjunction with the Spark processing engine. “Apache Parquet is an open source, column-oriented data file format designed for efficient data storage and retrieval” (Databricks, n.d.). Per our class discussion, the use of Parquet is a good tool for file compression and processing, although perhaps not the best.

While Parquet is commonly known to perform well within Spark, this may not be the case with other processing engines such as Hive. As an example for comparison, Ivanov, T., & Pergolesi, M. (2020) conducted an experiment to illustrate the performance of Parquet and ORC across various processing engines by measuring query execution times in seconds between the two data storage formats. As expected, Parquet generally performed best within the Spark engine. However, there were some exceptions to this performance, noting that certain query types such as HiveQL/Spark MLlib led to inconclusive behavior. Therefore, while Parquet can generally be assumed to perform best within Spark, there may be instances where query processing could unexpectedly take longer than expected.

Generally, ORC is used most with Hive. “Apache ORC is a columnar file format that provides optimizations to speed up queries. It is a far more efficient file format than CSV or JSON” (Databricks, n.d.). Hive, on the other hand, is a tool within Hadoop to act as a SQL engine for HDFS (Hadoop Distributed File System). ORC can also filter rows within the data. Since Hive is a tool that runs within Hadoop, and Hadoop instructs servers to operate on data that is stored locally, or on disk, ORC can run very effectively in this environment. Going back to the experiment from above, ORC generally accomplished the best performance with the Hive engine (Ivanov, T., & Pergolesi, M. (2020). To illustrate where this use-case would be effective, a hypothetical example could be where employees are categorized by department within an organization and have a performance score associated with their record. Various employees could be sent and stored on individual servers, or DataNodes within the organization’s data center, where calculations could then be performed on the data such as average performance score based on department. In this case, ORC data format could be effective because the data is all stored locally on disk across multiple servers.

The third data storage format is AVRO, which is often used with JSON file types. “Apache Avro is a compact binary data serialization format providing varied data structures” (Vohra, 2016). Unlike Parquet and ORC, AVRO is a row-based file format, and can write and therefore update data where the schema may change. Going back to the example above with our employee data, if the data is ingested via JSON files where one file has the three columns described above (employee name, department, performance evaluation score), and another has four columns because one of the employees has a hyphenated last name, or alias, AVRO would be an ideal choice. Expanding on this thought, AVRO is a good choice for bronze level data with respect to the Medallion Architecture. AVRO will not throw errors in the ETL (Extract, Transform, Load) process as easily as ORC or Parquet. However, for data that is not ingested in JSON format, AVRO is not the preferred choice due to its performance in comparison to ORC and Parquet.

**MapReduce Discussion**

MapReduce is a software framework that allows data processing in parallel across large clusters made up of many nodes. Although the tools used for MapReduce are different now as opposed to say, ten years ago, the concepts are mainly the same. There are a few major components of MapReduce. First, communication with the Name node occurs. Then, the mapper instructs tasks to be performed on the data across the cluster of data nodes. The tasks, whether it be filtering, conducting arithmetic calculations, or aggregating data would occur across the various data notes within the cluster. Typically, while this work is being performed, a ResourceManager (or RM) helps manage the tasks of data operations to be performed on the data nodes. The RM instructs the data nodes to spawn individual container(s) on each of the data node servers to regulate compute operations. After the tasks are complete, an ApplicationMaster (AM) communicates with the NodeManager on each server and back to the ResourceManager which then communicates with the edge node. From there, the reducer receives the key-value from the mapper and performs an operation to summarize the data. This summarization reduces the data from all the data nodes into an edge node, which is the same place where the mapping and direction to perform data tasks occur.

There is one major difference in the way MapReduce was first introduced in comparison to how it is commonly used now, which is the introduction of Cloud computing. According to Kazi (2024, as cited in Mell et al., 2015), “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” (p.102). When MapReduce was first introduced as part of the Hadoop toolset, the functionality was orchestrated by writing java code to scale across many machines in a cluster. All data and associated tasks were performed on disk, which meant the scanning, operations, and reading of data was all performed locally within the hardware on the cluster. With the introduction of Cloud computing, the MapReduce functionality can now be accomplished in memory. This functionality allows organizations to save time and resources through the reduction of on-premises hardware that is commonly prone to maintenance and failure. One of the most commonly used tools to achieve Cloud computing in this way is through Spark, which simplifies the MapReduce process by eliminating the need for Java in the Hadoop workflow.

MapReduce is critical to big data platforms such as Hadoop. Per Kazi (2024):

MapReduce is a programming framework, which organizes multiple computers in a cluster in order to perform the calculations needed. It takes care of distributing the work between computers and of putting together the results of each computer’s computation. Just as importantly, it takes care of hardware and network failures so that they do not affect the flow of computation. (p.99)

In the world of big data today, the amount of compute resources and calculations continues to expand. While data analysis and data science used to be plausible on a single machine, this is often no longer the case. For this reason, MapReduce is critical. Hardware failure is both possible and expected, and MapReduce ensures data redundancy across clusters. Furthermore, the volume of data contained across a cluster may very well be stored in terabytes, or even petabytes. A single machine with a single processor is not capable of feasibly processing this volume of data, and MapReduce spreads the workload across numerous machines in a cost-effective, timely fashion.

**Pros and Cons of Hadoop**

Hadoop offers tremendous advantages for organizations ingesting, processing, and making decisions from big data. First, Hadoop is very cost effective and scalable. I believe these two advantages are closely related. Since Hadoop is open-source, and operates on commodity hardware, it allows for organizations to divide large amounts of data across multiple inexpensive machine clusters (GeeksforGeeks, 2020). Additionally, Hadoop is flexible and allows for both structured and unstructured datasets, while also reducing network traffic by allowing the workload to be spread across multiple data nodes. By managing the workload across numerous machines, Hadoop allows for large amounts of data to be processed very quickly, while also providing data redundancy across clusters. To maximize efficiency, Hadoop takes advantage of parallel data processing, thus leading to a high throughput model where large amounts of tasks can get done very quickly.

While Hadoop offers many advantages, there are also disadvantages that should be mentioned. Hadoop is very much intended to be a tool to use when corresponding with large data. In fact, the efficiency of Hadoop generally decreases when working with small datasets, and often fails when it needs access to small files in a large amount less than 128MB (GeeksforGeeks, 2020). Additionally, Hadoop runs based on batch processes, which are not efficient and make a very poor choice for operations requiring low latency. Hadoop also presents multiple cybersecurity concerns, which are presented mainly in its underlying java framework and difficulty in managing Kerberos, which is the default security feature for Hadoop. In late 2021, a Log4J vulnerability allowed attackers to take advantage of java-based systems, which posed a thread for Hadoop systems implemented across many organizations. Finally, the underlying Hadoop framework requires calculations to be done from disk. Because of this, in-memory calculations are difficult to perform, which can lead to high overhead (GeeksforGeeks, 2020). This inefficiency has led to many organizations taking advantage of tools such as Spark, which perform in the cloud and can make in-memory calculations less inefficient.

All in all, the use of Hadoop can be very advantageous in certain situations and prove to be inefficient in others. As an example, if an advertising company collects user preferences to make decisions regarding desirable ad content, they would want to collect as much data as possible which could come in the form of free-text feedback amongst many others. Due to the large volume of information from many different places and in many different formats, Hadoop may be a great tool for performing data tasks. In contrast, an organization that acts as a defense contractor with the government may be more hesitant to use Hadoop due to its security vulnerabilities and encryption shortfalls. In conclusion, while Hadoop may be ineffective for some organizations possessing big data, it is incredibly important and critical for many to minimize costs and take advantage of efficient processing across entire clusters.

**References**

Databricks. (n.d.). *What is Parquet?* <https://www.databricks.com/glossary/what-is-parquet>

Databricks. (n.d.). *ORC File?* <https://docs.databricks.com/en/query/formats/orc.html>

Geeks for Geeks. (2020, June 29). *Hadoop – Pros and Cons.* <https://www.geeksforgeeks.org/hadoop-pros-and-cons/?ref=ml_lbp>

Ivanov, T., & Pergolesi, M. (2020). The impact of columnar file formats on SQL‐on‐hadoop engine performance: A study on ORC and Parquet. Concurrency and Computation: Practice and Experience, 32(5), e5523.

Kazi, S. A. (2024). Impact of Hadoop Technology in Cloud Computing. *Recent Advancements in Science and Technology*, 98.

Vohra, D. (2016). Apache avro. *Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools*, 303-323.