Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2019, 6(1):118-122

Research Article ISSN: 2394 - 658X

Optimizing Data Ingestion Frameworks in Distributed Systems

Chandrakanth Lekkala

Chan.Lekkala@gmail.com

ABSTRACT

As big data becomes prevalent, organizations need help gaining, processing, and absorbing large constellations of data coming from different sources. Distributed systems have taken the leading position in processing large amounts of data with complexity and volume. This article looks into optimizing data ingestion mechanisms on a dispersed system, advocating Apache Hadoop as a suitable option. Going into the Hadoop Distributed File System (HDFS) architecture and the significance of Map Reduce for data ingestion, we do so. Besides, we study various optimization techniques comprising data partitioning and parallel data loading, data compression and parallel computation to maximize the performance and scalability of data ingestion. We also cover the flaws and impediments that distributed data ingestion faces and offer a preview of how the research should move. Using optimized data ingestion methods, companies can effectively take the opportunities in the big data and measure the fruits of these analytics regarding business growth and innovation.

Key words: Data Collection, Distributed System, Big Data, Apache Hadoop, HDFS, Map Reduce, Performance, Scalability

INTRODUCTION

The fast-paced data growth in modern years has created explorative frontiers for organizations that want to harness big data as a basis for competitive advantage. The ability to process great volumes of data from a complex and multisource environment is critical for supercharging big data analytics. Distributed systems have become the de facto configuration for managing big data ingestion at the scale of complexity [1]. These systems contain the necessary infrastructure and frameworks to handle data storage, processing, and analysis distributed across systems with many nodes and the scalable and fault-tolerant pipes of data ingestion.

Apache Hadoop is embraced as an open-source framework for data ingestion and processing, specifically for distributed systems. The core building blocks Hadoop organizations use to handle big data, like the Hadoop Distributed File System (HDFS) and Map Reduce, have changed how data [2] is handled. HDFS supplies scalable and databases across nodes, while Map Reduce enables simultaneous processing of data in a distributed system.

Yet, overcoming challenges in data ingestion frameworks in the distributed system becomes essential. The huge volume of data and its velocity rate demand good data chunking methods to meet the balance of data splitting among nodes. Data compression approaches should be utilized to minimize storage and network overheads. It is also critical to perform data fetching in parallel for faster data ingestion. Additionally, scalability issues, including horizontal and vertical scaling, must be considered to cope with rising data volume and processing demand.

In this essay, we investigate a problem of data ingestion optimization in distributed systems with Hadoop as our choice of implementation. We then cover HDFS architecture and Map Reduce related to Data Ingestion. We examine multiple optimization strategies, including database partitioning, data compression, and parallel loading, to provide a robust data ingestion process with better efficiency and scalability. In addition, we examine the issues and barriers that distributed data ingestion presents and suggest the focus for future research directions.

PROBLEM STATEMENT

Data explosion is the fastest-growing problem that has eased the way organizations take action to get some benefit from big data for competitive advantage. The power of fast integration and digestion of massive amounts of data from different sources is key to appropriately using the data analysis opportunity. Traditional data intake tools are often overwhelmed by the scale, variety, and pace of big data that, in turn, results in the impasse of performance and scaling out [3].

A distributed system has become a compelling method of resolving the big data ingestion issue. Data and instruction distribution among multiple XMLs leads to scalable and fault-tolerant data pipelines. Nevertheless, finding the precise data ingestion pattern in distributed systems among these options is complex. Employing proper data partition methods with compression techniques and parallel data loading mechanisms and considering the system's scalability is very important [4].

BIG DATA INTAKE IN BIG DATA SYSTEMS

Data ingestion implies obtaining, importing and processing data from different available sources into one storage place within the target system or repository [5]. The data ingestion process occupies a central position, with big data helping organizations grasp and use immense amounts of structured, semi-structured, and unstructured information.

Generally, different types of data are involved in the big data ecosystem, i.e., transactional systems, social media, sensor networks, log files, and streaming data [6]. Large numbers and heterogeneous data sources online are daunting for data ingestion architectures. Regular data integration approaches, i.e. extract, transform, and load (ETL) processes, are also challenged by the volume and speed of the big data.

These problems have given rise to distributed data ingestion frameworks, which play a pivotal role in nearly all big data environments. These approaches, among others, exploit the power of distributed computing to break down large data ingestion tasks into finer pieces that can be parallelized across multiple nodes for scalable and efficient data processing [7].

Apache Hadoop is now a trendy open-source framework used for data collection and processing in a distributed way. Through Hadoop, we can store and process large volumes of data across clusters of inexpensive commodity hardware, ensuring they are scalable and tolerant of failures. The main parts of Hadoop, like HDFS (HDFS) and Map Reduce, have created a new direction in organizations' big data intake and processing.

DISTRIBUTED APACHE HADOOP IS DATA INGESTION

HDFS Architecture

HDFS, a distributed file system, is a core component of the Hadoop ecosystem of Apache Hadoop. Hadoop Distributed File System, or HDFS, is designed to handle large volumes of datasets with distributed deployment among commodity hardware clusters. It allows data to be stored at scale and fault-tolerantly, which makes it a perfect fit for big data ingestion and processing. HDFS is the so-called master-slave architecture consisting of the Name Node and a bunch of Data Nodes. Here, the Name Node operates as the controller node and takes charge of namespaces and meta-data management. It keeps the directory structure and manages the data locations over the Data Nodes. Unlike the Miners, Data Nodes, as their names imply, are the nodes responsible for storing and retrieving data blocks [11].

The Info HDFS divides data into fixed-size blocks, typically 64MB or 128MB, and distributes them on Data Nodes. Every block is duplicated across multiple data nodes to ensure fault tolerance and data availability. The replica was configurable, and the number of copies of each block saved in the cluster was determined [11].

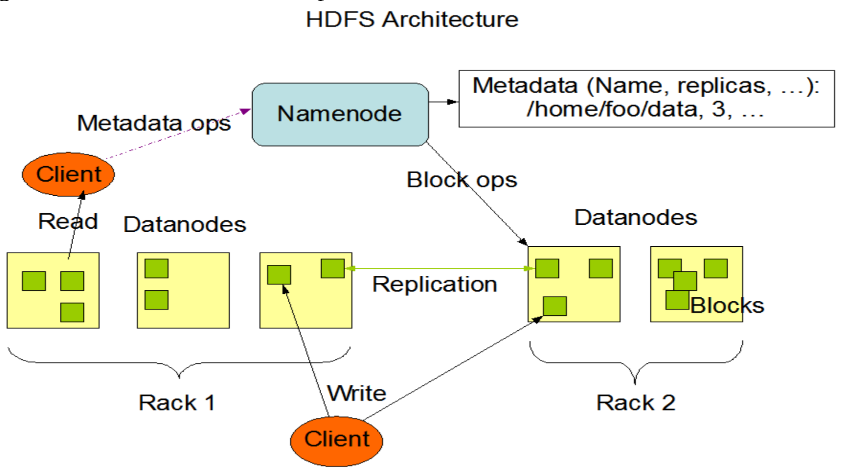

Figure 1: Exhibits the HDFS structure, emphasizing the interaction between the Name Node and the Data Nodes

Hadoop for Data Ingestion with Map Reduce
Map Reduce is the programming model and the processing framework that allows the distributed processing of large volumes of data with the Hadoop system. Its scalability and fault tolerance features provide a means of reading and analyzing data in batches.
The Map Reduce framework consists of two main phases: the Map and Reduce processes. In the Map phase, initial data is divided into chunks and processed in parallel by multiple nodes. On the other hand, the map task works on a subset of the data, giving out key-value pairs as intermediary output. The Reduce phase is when the intermediate key-value pairs merge, sort, and sum up based on the keys [10].
Map Reduce can leverage parallelization techniques for these data ingestion operations, such as preprocessing, filtering, and aggregation. It enforces the parallel data processing of large streaming data, thus scaling and increasing the throughput of your data.

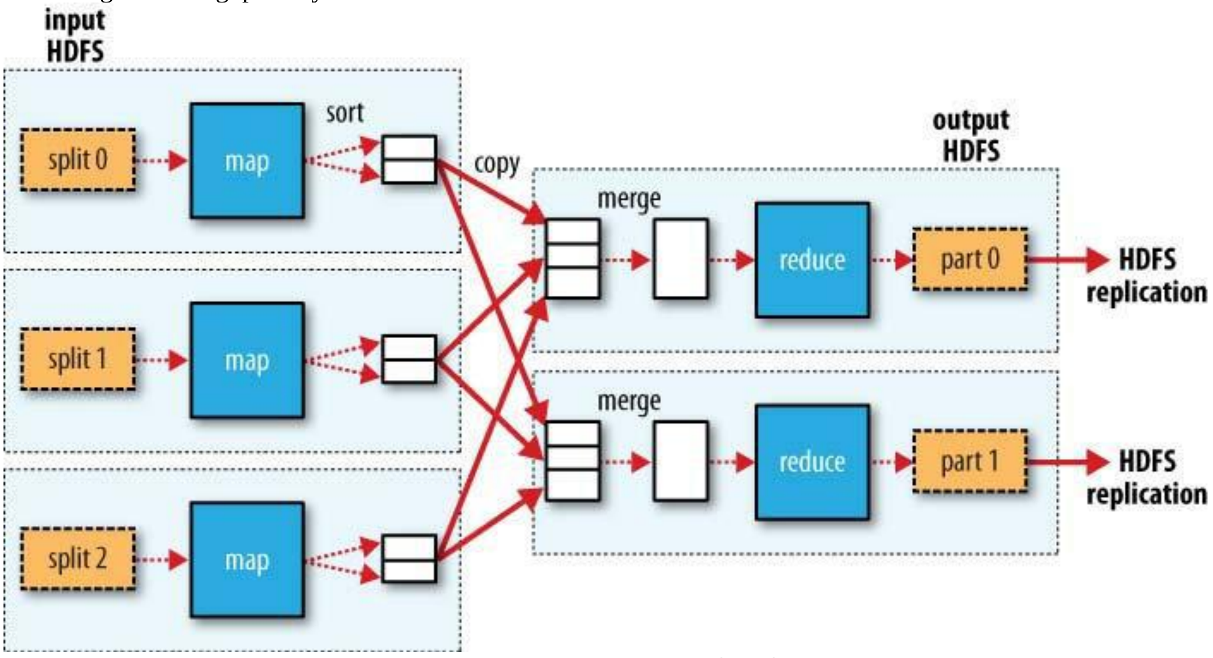

Figure 2: Illustrates the Map Reduce data flow for data ingestion

OPTIMIZING DATA INGESTION PERFORMANCE

Data Partitioning Strategies
Partitioning data is a critical consideration to boost ingestion performance for distributed systems. Implementing well-suited partitioning approaches guarantees that the data will be evenly distributed throughout the cluster nodes, which is beneficial for running a parallel code and efficient resource utilization [14].
One of the most conventional techniques of sub-dividing the data depending on a specific key or attribute is to use hash partitioning, in which data is allocated based on the hash function applied to a particular key or attribute. Hash partitioning is a function that will even the data distribution across partitions, resulting in lower data skew and bringing the load balancing of the Hadoop cluster [15].
The second divide pattern is range partitioning, wherein data is broken into partitions of values across a specified range. The partitional technique is useful when there is a natural ordering of data or a particular range of data is commonly used together.
Compression Techniques
Compression technology is at the heart of maximizing data input performance by minimizing storage space needed and network bandwidth utilized. Data compression before data collection reduces the amount of data transferred through the network and stored in distributed file systems [17].
Hadoop supports numerous compression codecs, including Snappy, LZO, and Gzip, which offer speed and storage dealing with balance. Snappy often utilizes RAM to offer high-speed decompression and compression. Thus, it can be used for real-time data ingestion [18].
Parallel Data Loading
Side-by-side data loading processes are required to get the maximum output for data streaming in the distributed infrastructure. This way, Hadoop can distribute the data across different nodes and thus become faster compared to other technologies [19].
Hadoop has various instruments and frameworks for parallelly loading data in the volume; for example, the DistCp (Distributed Copy) tool performs a copy of data between Hadoop clusters or from external data sources

to HDFS [20]. Furthermore, the Map Reduce framework can be employed by running a custom Map Reduce job (e.g. for parallel data ingestion from various sources and writing it to HDFS).

Scalability Considerations

Scalability is the key point when creating data intake frameworks in distributed systems. Increasing the data volumes means that the system should be able to scale horizontally by adding more nodes to the cluster and vertically by enhancing the resources of individual nodes [21].With Hadoop, horizontal expandability is provided via the ability to attach new data nodes to the cluster at a time. However, this ensures organizations prevent downtime or performance degradation, as they may be required to meet growing data volumes or processing requirements.

On the other hand, vertical scaling stands for increasing an individual node's resources. I.e. a node's memory, CPU and storage space. Hadoop offers YARN (Yet another Resource Negotiator) that contributes to dynamic resource allocation and management, leading to a high cluster resource usage level [22].

Challenges and Limitations

Although distributed data ingestion platforms such as Apache Hadoop provide enormous benefits, you should analyze and deal with the limitations and challenges.

Data conflicts and their consistency and reliability emerge as the key issues in distributed systems. This is where data integrity and consistency across multiple nodes appears, so carefully managed data replication, fault-tolerance, and data validation mechanisms are needed.

The security and privacy of the given data are also important factors, especially when dealing with sensitive or confidential data. Data safety during ingestion and storage is vital; therefore, security measures should be installed, such as encryption, access controls and data masking [24].

Another challenge is that the pipelines for defining distributed data are complicated to manage and control. It could take a significant amount of time to debug and solve a problem in a distributed environment and be challenging. This requires appropriate monitoring and logging processes where bottlenecks and other problems can be detected and resolved to guarantee data ingestion works seamlessly [25].

CONCLUSION

Ensuring the efficiency of data ingestion structures in distributed systems is fundamental for businesses that intend to gain the greatest benefits from big data. Apache Hadoop and innovative organizations such as HDFS and Map Reduce can be used to store and process data for clusters. Companies can greatly improve the performance and scalability of their data onboarding channels by using data-partitioning best practices, compression techniques, and parallel data load mechanisms. Nevertheless, it is also necessary to keep track of data consistency, security issues, and data management complexity.

Taking these studies forward, the aim is to devise better data partitioning algorithms, additional data compression methods, machine learning, and artificial intelligence integration for smart data ingestion and processing. On top of that, the issue of data consistency, privacy, and management of a distributed system will be vital for the successful adoption and deployment of optimized data ingestion frameworks.

What has greater volume, diversity, and velocity of data, data ingestion in the distributed system will remain in the area of research that is getting the most attention and innovation. Utilizing the distributed computing of 'Big Data' and the development of data ingestion optimization, organizations get access to the amazing power of data and start extracting business value from it using data-driven insights and decision-making.

REFERENCES

[1]. Dean, J. and Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, *51*(1), pp.107-113.

[2]. White, T., 2012. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.".

[3]. Sakr, S., 2016. *Big data 2.0 processing systems: a survey* (Vol. 2142). New York: Springer.

[4]. Sumbaly, R., Kreps, J. and Shah, S., 2013, June. The big data ecosystem at linkedin. In *Proceedings of the 2013 acm sigmod international conference on management of data* (pp. 1125-1134), *105*, pp.250-259.

[5]. Banu, A.B. and Nivedita, V.S., 2019. Trending Big Data Tools for Industrial Data Analytics. In *Encyclopedia of Data Science and Machine Learning* (pp. 545-565). IGI Global.

[6]. Gandomi, A. and Haider, M., 2015. Beyond the hype: Big data concepts, methods, and analytics. *International journal of information management*, *35*(2), pp.137-144.

[7]. Zaharia, M., Xin, R.S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M.J. and Ghodsi, A., 2016. Apache spark: a unified engine for big data processing. *Communications of the ACM*, *59*(11), pp.56-65.

[8]. Shvachko, K., Kuang, H., Radia, S. and Chansler, R., 2010, May. The hadoop distributed file system. In *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)* (pp. 1-10). Ieee.

[9]. Borthakur, D., 2008. HDFS architecture guide. *Hadoop apache project*, *53*(1-13), p.2.

[10]. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S. and Stoica, I., 2010. Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*.

[11]. T. White, "The Small Files Problem," in Hadoop: The Definitive Guide, 4th ed., O'Reilly Media, 2015, pp. 83-85.

[12]. Mason, R. (2013). Using communications media in open and flexible learning. Routledge.

[13]. Ma, M., Wang, P., & Chu, C. H. (2013, August). Data management for internet of things: Challenges, approaches and opportunities. In 2013 IEEE International conference on green computing and communications and IEEE Internet of Things and IEEE cyber, physical and social computing (pp. 1144-1151). IEEE.

[14]. Saha, B., Shah, H., Seth, S., Vijayaraghavan, G., Murthy, A. and Curino, C., 2015, May. Apache tez: A unifying framework for modeling and building data processing applications. In *Proceedings of the 2015 ACM SIGMOD international conference on Management of Data* (pp. 1357-1369).

[15]. Agarwal, S., Mozafari, B., Panda, A., Milner, H., Madden, S. and Stoica, I., 2013, April. BlinkDB: queries with bounded errors and bounded response times on very large data. In *Proceedings of the 8th ACM European conference on computer systems* (pp. 29-42).

[16]. Kornacker, M., Behm, A., Bittorf, V., Bobrovytsky, T., Ching, C., Choi, A., Erickson, J., Grund, M., Hecht, D., Jacobs, M. and Joshi, I., 2015, January. Impala: A Modern, Open-Source SQL Engine for Hadoop. In *Cidr* (Vol. 1, p. 9).

[17]. Chen, Y., Alspaugh, S. and Katz, R., 2012. Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. *arXiv preprint arXiv:1208.4174*.

[18]. Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., & Pras, A. (2014). Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials*, *16*(4), 2037-2064.

[19]. Ghazal, A., Rabl, T., Hu, M., Raab, F., Poess, M., Crolotte, A. and Jacobsen, H.A., 2013, June. Bigbench: Towards an industry standard benchmark for big data analytics. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data* (pp. 1197-1208).

[20]. Chiba, T. and Onodera, T., 2016, April. Workload characterization and optimization of tpc-h queries on apache spark. In *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)* (pp. 112-121). IEEE.

[21]. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H. and Murthy, R., 2010, March. Hive-a petabyte scale data warehouse using hadoop. In *2010 IEEE 26th international conference on data engineering (ICDE 2010)* (pp. 996-1005). IEEE.

[22]. Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S. and Saha, B., 2013, October. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing* (pp. 1-16).

[23]. Shvachko, K.V., 2010. HDFS Scalability: The limits to growth.*; login:: the magazine of USENIX & SAGE*, *35*(2), pp.6-16.

[24]. Keshvadi, S. and Faghih, B., 2016. A multi-agent based load balancing system in IaaS cloud environment. *International Robotics & Automation Journal*, *1*(1), pp.1-6.

[25]. Mavridis, I. and Karatza, H., 2017. Performance evaluation of cloud-based log file analysis with Apache Hadoop and Apache Spark. *Journal of Systems and Software*, *125*, pp.133-151.