Dual ETL – Hadoop Cluster Auto Failover

Sainath Muvva

Abstract

This paper examines the design of data infrastructure for high-speed delivery, focusing on the 4 V's of big data and the importance of geographically separated primary and Disaster Recovery clusters. It explores the complexities of the failover process of Hadoop clusters, identifying challenges such as manual metadata updates and data quality checks. The research proposes automation solutions, including the use of DistCp for data replication and Hive commands for metadata updates, aiming to enhance data infrastructure resilience and reduce manual intervention during critical events.

Keywords: **ETL, Hadoop, Distcp, Data Quality**

Introduction

In the era of data-driven decision-making, organizations are prioritizing high-speed data delivery to stakeholders. This paper explores critical aspects of data infrastructure design, focusing on the 4 V's of big data: velocity, veracity, variety, and volume [1]. We examine the common practice of maintaining geographically separated primary and Disaster Recovery (DR) clusters in on-premises data storage solutions, highlighting its importance for business continuity.

The study delves into the complexities of the failover process with a focus on Hadoop clusters, a crucial mechanism for maintaining operations during primary cluster downtime. We identify challenges associated with failovers, including manual tasks such as metadata updates and data quality checks. The paper then proposes methods for automating this process, specifically discussing the application of DistCp (Distributed Copy) for inter-cluster data replication and Hive commands for table metadata updates.

By addressing these challenges and exploring automation solutions, this research aims to contribute to the development of more efficient and reliable failover procedures, thereby enhancing overall data infrastructure resilience and reducing the need for manual intervention during critical events.

Hadoop

Apache Hadoop is developed on the principles of distributed systems to address the major challenges posed by big data [2]. As web media generates vast amounts of data on a daily basis, it has become increasingly difficult to process, store, and analyze this information. Apache Hadoop is an open-source framework for distributed computing and storage of unstructured data using commodity hardware. Hadoop runs applications using MapReduce, a programming model that processes data in parallel.

The Hadoop Ecosystem comprises various tools that complement each other to achieve desired outputs. MapReduce and HDFS (Hadoop Distributed File System) are two key components primarily responsible for data processing and storage, respectively. Figure 1 depicts the tools within the Hadoop ecosystem. These tools are instrumental in deriving desired statistical and analytical outcomes from processed big data.

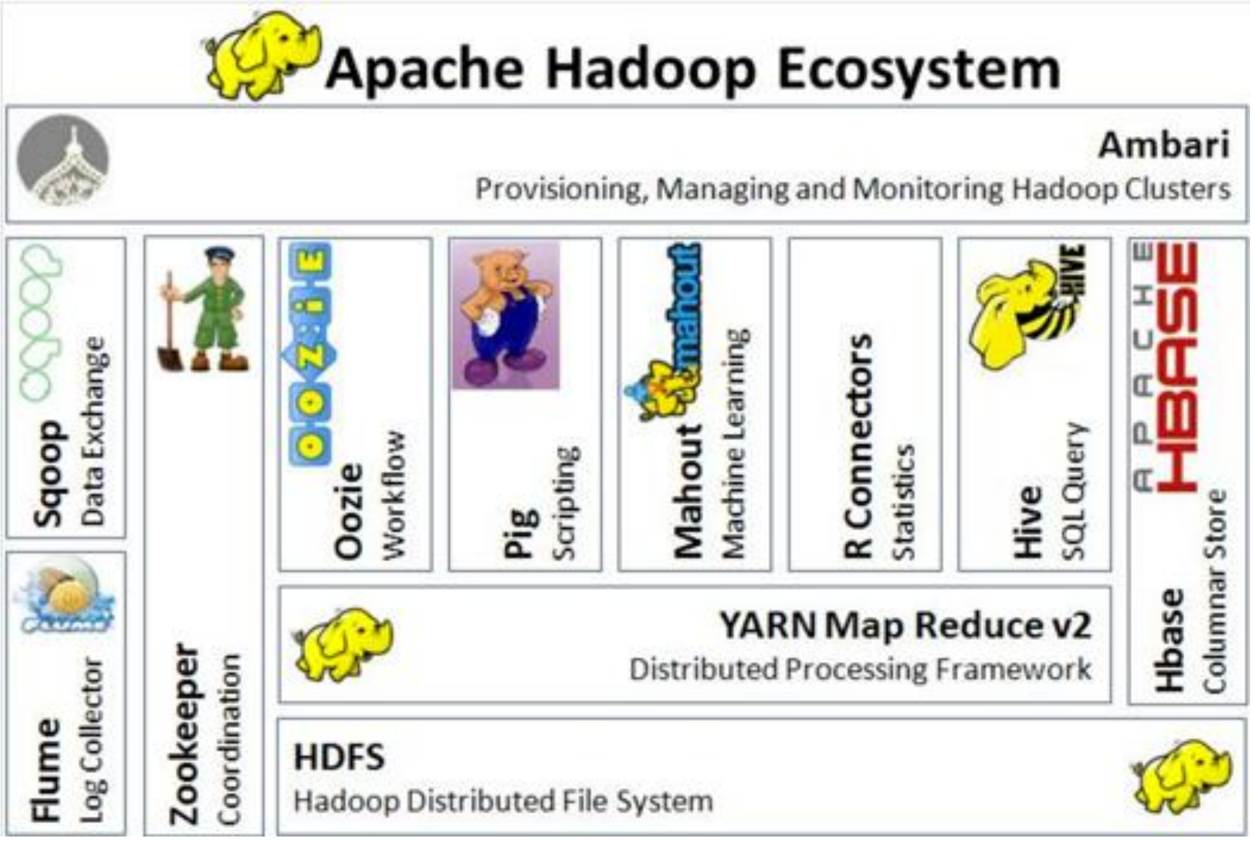

Fig1. Hadoop Ecosystem

Hive

Apache Hive is an open-source data warehouse system for analytics on big-data workloads. It is a powerful tool within the Hadoop ecosystem, primarily focused on ETL (Extract, Transform, Load) processes and batch reporting workloads. Hive is capable of reading huge volumes of data, transforming this data, and loading it into Hive tables that are internally stored in HDFS(Hadoop Distributed File System) [3].

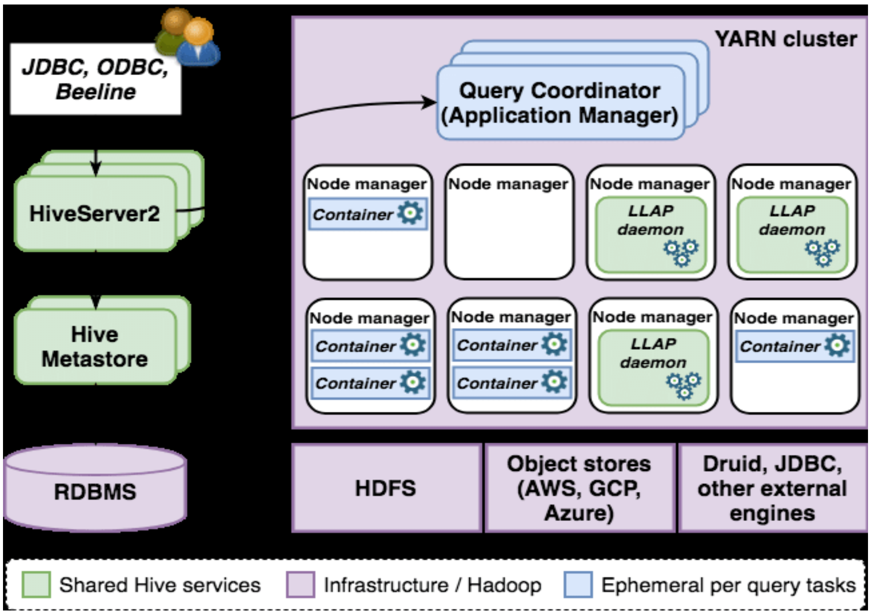

Fig 2. Hive Architecture

Distcp

DistCp (Distributed Copy) is a tool for copying HDFS files either from one Hadoop cluster to another or within the same cluster. It is similar to a regular copy job, with the key difference being its parallel nature, designed for bulk copying operations. DistCp uses the MapReduce framework to perform these copy operations [4].

Sample distcp command with checksum: hadoopdistcp -update {src_path}{destination_path}

To skip checksum validation: hadoopdistcp -skipcrccheck -update srcdest

Dual ETL for Auto Failover

Many organizations operating Hadoop clusters within their data centers employ a periodic data replication strategy from their primary cluster to a Disaster Recovery (DR) cluster. Downstream processes, including report generation, data science models, and data visualization, typically depend on the primary cluster's data. However, during unexpected downtime of the primary cluster, all upstream, downstream, and semantic jobs must transition to the DR cluster. This failover process necessitates manual synchronization checks between the primary and DR clusters, covering aspects such as code updates, data freshness, and other critical factors. This approach is often time-consuming and susceptible to errors.

To streamline this process and ensure seamless failover, a Dual ETL system has been implemented. This system eliminates the traditional primary/DR cluster distinction, instead maintaining two continuously operational clusters. Each cluster initiates ETL or data processing once its dependencies are met. Upon completion, the faster cluster logs an entry in a shared SQL audit table. The other cluster, running a parallel ETL process for the same reporting period, consults this audit table. If it finds a corresponding entry (indicating the other cluster has progressed further), it uses distcp to retrieve the data from the advanced cluster and updates its Hive metadata using the 'MSCK REPAIR TABLE {table_name}' command.

Conversely, if no entry is found, it signifies that the current cluster has completed the job first. Downstream applications also query this SQL audit table to determine which cluster to use. The cluster that finishes processing first becomes the de facto primary for that particular run, with all dashboards and reporting jobs directed to it. This dynamic primary designation can shift between clusters based on processing speed in subsequent runs, ensuring that downstream jobs always point to the most up-to-date data source.

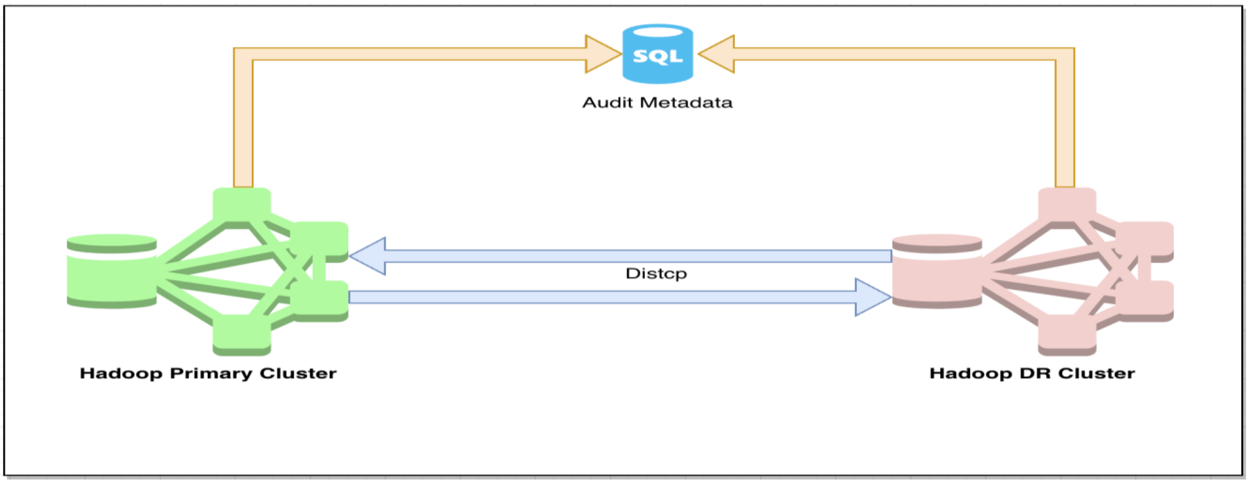

Fig 3. Dual ETL Architecture

Pros:

- **Always online and in sync**: Both systems are continuously operational and synchronized, with data replication and checksum validation through distcp after each run, eliminating the need for manual Data Quality checks.
- **Improved SLA**: Data is exported from the cluster that finishes processing earlier, leading to better Service Level Agreements.
- **Less error-prone**: Manual intervention is avoided, reducing the likelihood of human errors.

Cons:

- Resource utilization: Resources are used in both clusters, which could be seen as inefficient or costly.
- Debugging challenges: It may be more difficult to determine the source of exported data, requiring reference to the Audit table.

Challenges

Key Challenges in the Design:

▪ SQL Audit Metadata Table Availability:

• Risk: Downtime in the hosting data center (DC) renders the table inaccessible.

• Solution: Replicate the small metadata table to other DCs, ensuring quick and easy replication.

▪ Distcp Checksum Limitations:

• Constraint: Effective only when primary and distcp clusters have similar hardware and identical block sizes.

• Impact: Restricts flexibility in hardware choices and upgrades.

• Additional limitation: Checksum functionality doesn't work with encrypted files.

▪ Namenode Failure Vulnerability:

• Risk: Hard failure of either cluster's Namenode causes distcp process failure.

• Consequence: Potential disruption of the entire replication process due to this single point of failure.

▪ Necessity for Widespread Adoption:

• Requirement: Optimal performance of Dual ETL depends on broad implementation across upstream and downstream applications.

• Design Implication: Framework must accommodate diverse use cases to encourage widespread adoption.

To address these challenges, implementing robust exception handling in the auto-failover framework is crucial. This should include strategies for dealing with metadata inaccessibility, hardware discrepancies, Namenode failures, and ensuring compatibility with various application requirements. Comprehensive monitoring and alert systems are also essential for prompt issue detection and resolution.

Conclusion

A well-designed and properly implemented Dual ETL system offers substantial advantages during failover scenarios:

1. Automated Failover Management: It significantly reduces the need for manual intervention, streamlining the failover process.
2. Data Consistency: The system maintains data integrity across clusters, ensuring minimal discrepancies during and after failovers.
3. SLA Adherence: By automating failover procedures and preserving data consistency, it helps organizations meet their Service Level Agreements more reliably.

4. Operational Efficiency: The reduction in manual effort not only saves time but also minimizes the risk of human error during critical failover events.

In summary, Dual ETL transforms failover management from a potentially disruptive, labor-intensive process into a more seamless, automated operation. This approach not only alleviates the burden on IT teams during high-stress situations but also enhances overall system reliability and performance continuity.

Reference

1. BlagojRistevski "Hadoop as a Platform for Big Data Analytics in Healthcare and Medicine" https://www.researchgate.net/publication/338516812_Hadoop_as_a_Platform_for_Big_Data_Analytics_in_Healthcare_and_Medicine(accessed Aug. 17, 2019)
2. Ashlesha S. Nagdive, R. M. Tugnayat"A Review of Hadoop Ecosystem for BigData", https://www.ijcaonline.org/archives/volume180/number14/nagdive-2018-ijca-916273.pdf(accessed Aug. 10, 2019)
3. Jesús Camacho-Rodríguez, Ashutosh Chauhan, Alan Gates, Eugene Koifman, Owen O'Malley, Vineet Garg, Zoltan Haindrich, Sergey Shelukhin, Prasanth Jayachandran, Siddharth Seth, Deepak Jaiswal, Slim Bouguerra, Nishant Bangarwa, Sankar Hariappan, Anishek Agarwal, Jason Dere, Daniel Dai, Thejas Nair, Nita Dembla, Gopal Vijayaraghavan, Günther Hagleitner "Apache Hive: From MapReduce to Enterprise-grade Big Data Warehousing", https://arxiv.org/pdf/1903.10970 (accessed July. 30, 2019)
4. https://hadoop.apache.org/docs/r2.8.5/hadoop-distcp/DistCp.html(accessed Aug. 15, 2019).