Migrating Hadoop Data to Google Cloud

Sainath Muvva

Abstract:
As companies are generating data at an unprecedented rate, the growing demand for increased computing power, storage capacity, and data governance is compelling them to transition from on-premises infrastructure to cloud-based solutions. This migration to the cloud has gained significant momentum in recent years, and the trend is likely to continue. The three major cloud service providers are AWS, Microsoft Azure, and Google Cloud Platform (GCP).
This paper serves as a guide for migrating data stored on the Hadoop File System to Google Cloud Storage buckets and Google BigQuery. It also discusses Google's Dataproc, which is the equivalent of Hadoop in the GCP ecosystem. The paper addresses the challenges, cost-efficiency considerations, and data governance aspects involved in this migration process to the Google Cloud Platform.

Keywords: Hadoop, GCP, Dataproc, BigQuery, Parquet

Introduction

Businesses are treating data as a highly valuable asset and are investing heavily in storage and computing solutions to handle the tremendous growth in data generation. This rapid increase in data volume has created unique challenges in scaling on-premises infrastructure, such as expanding data centers, maintaining cluster upgrades, and replacing faulty nodes. Cloud services, in the form of Software-as-a-Service (SaaS), have emerged as a solution. This has enabled companies to move away from managing physical server systems housed in less-than-ideal facilities, and instead leverage public cloud providers [1]. The question arises - why do organizations continue to spend time, effort, and money maintaining their own systems and servers when cloud providers can offer these services, allowing the organizations to focus on their core business and drive profitability? The key challenge in migrating to the cloud is moving data off on-premises systems and into the cloud. Cloud providers do offer services to help migrate databases from on-premises to the cloud. However, when it comes to data stored in the Hadoop File System, there is no easy way to accomplish this migration, and there is limited documentation explaining the process.
This paper discusses an efficient and high-speed approach for migrating historical data stored in the Hadoop Distributed File System (HDFS) to Google Cloud Storage (GCS) buckets. Although the paper focuses specifically on GCS, the concepts presented can be applied to migrating data to other cloud providers as well. This is because the tool referenced in the paper for performing the migration is a default component included with Hadoop distributions. The paper explains how Dataproc jobs are utilized to create Hive tables and update the Hive metadata for the copied data files. Additionally, the paper covers how the data migrated to GCS is then uploaded to Google BigQuery, making the migrated data queryable and accessible in BigQuery.

Hadoop File System Fundamentals

HDFS, or the Hadoop Distributed File System, stores data in a distributed way across multiple nodes in a cluster (a set of multiple machines connected over a LAN or MAN connection). There are two main components in Hadoop: the NameNode and the DataNode. The NameNode stores the metadata like the name, path, and other information about the files. The DataNodes store the actual data. When a human interacts with the Hadoop cluster, the NameNode initiates a block-level operation to generate a query to find the location of the file on the HDFS. Since all data is stored in blocks, the NameNode retrieves the block information and shares it.

For write operations, the process is similar, but instead of querying, the NameNode writes the data to the HDFS. This is done only once. By default, the Hadoop replication factor is 3, which means the same data file is replicated to 3 different DataNodes located on different racks. This is to enable data recovery in case one node goes down [2].

Example: hdfs://{NameNode}/{file_path}

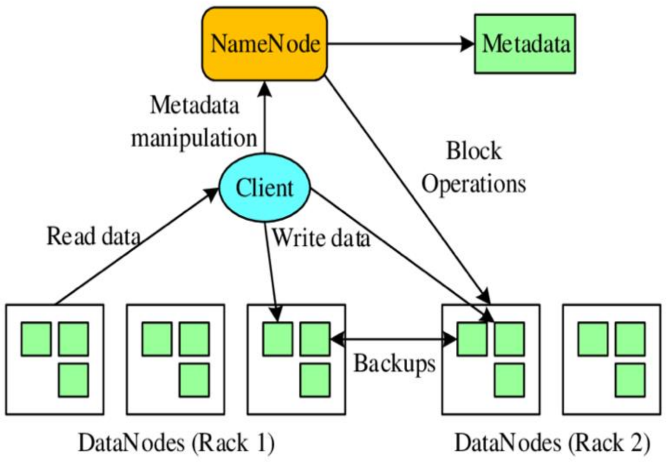

Figure 1: Hadoop File System Architecture

Google Cloud Storage

Google Cloud Storage is a managed service by Google for storing structured, semi-structured and unstructured data. It can store any amount of data and is retrievable whenever needed.

Example: gs://{bucket_name}/{file_path}

Tool and Framework

DistCp, or Distributed Copy, is a tool used for copying data files between or within a Hadoop cluster. It uses the MapReduce framework for copying the files, and it has error handling, recovery, and reporting capabilities. DistCp reads the list of files and folders as input to the mapper tasks, and each mapper copies those folders or files to the destination folder [3]. The Hadoop distcp command can be executed through Command line interface

Example: **hadoop distcp hdfs://{source_name_node}/foo/bar \
hdfs://{target_name_node}/bar/foo**

Hadoop DistCp can be used to copy data from HDFS to Cloud Storage. There are two migration models: push and pull. In the push model, which is simpler, the source cluster runs the distcp jobs on its data nodes and pushes files directly to Cloud Storage. In the pull model, which is more complex but offers several advantages, an ephemeral Dataproc cluster runs the distcp jobs on its data nodes, pulls files from the source cluster, and copies them to Cloud Storage. The pull model minimizes impact on the source cluster's resources, reduces network traffic, and doesn't require installing the Cloud Storage connector on the source cluster. The only requirement is to establish a private link between your on-premises network and Google's network using Cloud Interconnect or Cloud VPN before using DistCp.

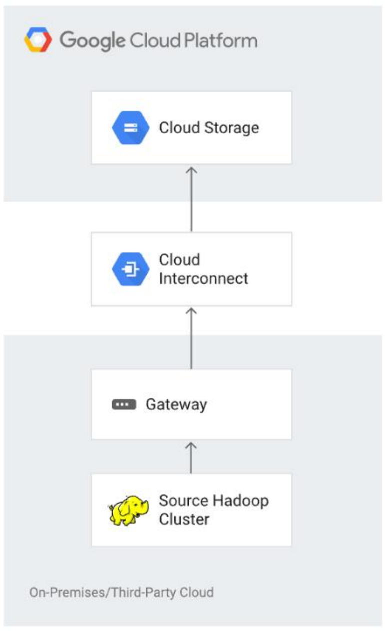

Figure 2: Push Model

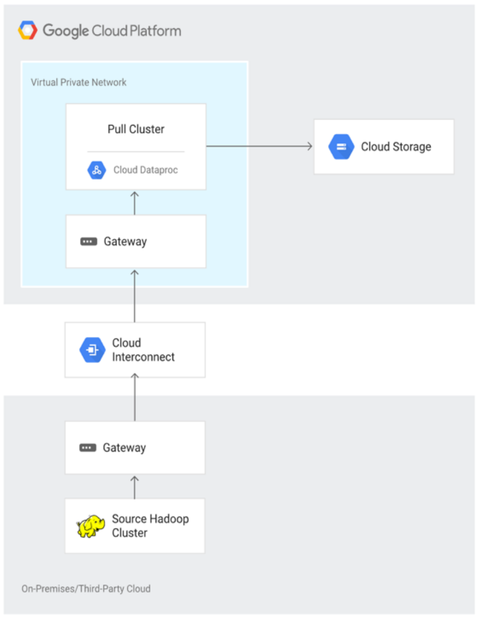

Figure 3: Pull Model

Push Model:

Pros:

- Simplest model to implement
- No need to create extra compute resources, so is less expensive compared to pull model

- Direct push from source cluster to Cloud Storage

Cons:
- Can impact source cluster's resources (CPU, RAM, network bandwidth)
- May interfere with regular data processing jobs on source cluster
- Higher network traffic (up to twice the file's total size)

Pull Model:
Pros:
- Minimizes impact on source cluster's CPU and RAM resources
- Reduced traffic on source cluster's network
- Higher outbound bandwidths and faster transfers
- No need to install Cloud Storage connector on source cluster
- Network traffic is limited to the file's total size
- Can fine-tune pull cluster's resources on Google Cloud
- Can tear down pull cluster after migration

Cons:
- More complex to implement than push model
- Requires creation of additional compute resources (ephemeral Dataproc cluster)
- Needs monitoring of distcp map tasks and bandwidth to avoid overwhelming source cluster

Example: **hadoop distcp -m 400 hdfs://{on-prem Name_node}/{file_path} gs://{bucket_name}/{file_path}**

The above command can be incorporated into a framework that recursively copies files in a multithreaded fashion, optimizing performance based on available network bandwidth and cluster resources where DistCp is executed.

Once the files are copied, a Dataproc [4] cluster is initiated. An external table is then created within the Hive metastore [5], and the partitions are updated using the Hive command 'MSCK REPAIR TABLE {table_name};'. This process updates the Hive table metadata, ensuring it remains synchronized with the on-premises data. These tables can be sourced for running hive or spark jobs to derive business insights from the data.

Sample dataproc job to create an external table

gcloud dataproc jobs submit hive --cluster={dataproc_cluster_name} -e="CREATE EXTERNAL TABLE {migratied_table}(bar int) LOCATION 'gs://{ gcs_bucket_with_copied_files}/{data_folder_path}'" -e="'MSCK REPAIR TABLE { migratied_table }"[9]

BigQuery

BigQuery [7] is a popular data analytics platform provided and managed by Google. It supports multiple file formats like Parquet, ORC, CSV, and others. It is known for its fast data retrieval speeds. While Dataproc is used for batch data processing, BigQuery is used for analytics thanks to its ease of use and minimal query response times.

BigQuery has API's [8] that supports multiple languages for various use cases. Below code snippet is for uploading the parquet files to Google BigQuery from Google cloud storage

```
job_config = bigquery.LoadJobConfig(
   write_disposition=bigquery.WriteDisposition.WRITE_TRUNCATE,
   source_format=bigquery.SourceFormat.PARQUET,
)
uri = "gs://{gcs_bucket_with_copied_files}/{parquet file path}"
load_job = client.load_table_from_uri(
   uri, table_id, job_config=job_config
)  # Make an API request.

load_job.result() [8]
```

As parquet files have embedded schemas, BigQuery can intelligently read the schema and create the table accordingly, including appropriate partitions. However, BigQuery will use default data distribution unless specified otherwise.

Challenges

Some common mistakes in migrating data from on-premises systems to Google Cloud Storage (GCS) involve checksum validations supported by DistCp. These validations often fail when copying files from on-premises to GCS due to differences in block sizes between on-premises HDFS data nodes and GCS. Another challenge is executing too many DistCp jobs concurrently, which can exhaust resources and delay the migration process.

As this migration involves multiple hops (HDFS → GCS → Dataproc/Hive → BigQuery), tracking the status of all datasets throughout the migration can be difficult. It is recommended to maintain an audit table to track the stages of migration at the dataset level.

Here's a revised and improved version of the sentence:

When making concurrent calls to the BigQuery API or submitting multiple Dataproc jobs simultaneously, intermittent failures may occur due to exceeding the API's concurrent request threshold. To mitigate this issue, it is recommended to implement a retry mechanism in the framework.

Conclusion

This paper covers the migration of data from HDFS to Google Cloud Platform (GCP), focusing on the services of Google Cloud Storage (GCS), Dataproc, and BigQuery. While this approach is specific to GCP, DistCp can be used for migrating data to other platforms as well. A similar approach can also be implemented for incremental, day-to-day data copying if needed. In the future, Google may develop a managed service that copies data into different services, potentially eliminating the need for this approach. However, for now, this method provides the best-suited, scalable, and fault-tolerant solution capable of copying terabytes to petabytes of data within a few days.

References:

1. Jacek Materna. https://www.forbes.com/sites/forbestechcouncil/2018/08/13/on-premise-is-dead-long-live-on-premise/ (accessed Oct. 11, 2019).
2. Neeta Sharma, https://www.researchgate.net/publication/329586205_Enhancing_the_Traditional_File_System_to_HDFS_A_Big_Data_Solution (accessed Oct. 11, 2019).

3. https://hadoop.apache.org/docs/r2.8.5/hadoop-distcp/DistCp.html(accessed Oct. 13, 2019).
4. https://cloud.google.com/architecture/hadoop/hadoop-gcp-migration-data(accessed Oct. 15, 2019).
5. https://cloud.google.com/dataproc?hl=en(accessed Oct. 15, 2019).
6. https://cloud.google.com/dataproc-metastore/docs/hive-metastore(accessed Oct. 16, 2019).
7. https://cloud.google.com/bigquery?hl=en(accessed Oct. 19, 2019).
8. https://cloud.google.com/bigquery/docs/loading-data-cloud-storage-parquet(accessed Oct. 19, 2019).
9. https://cloud.google.com/sdk/gcloud/reference/dataproc/jobs/submit/hive(accessed Oct. 21, 2019).