VAST S3 Storage Data Platform User Guide

Contents

1 VAST S3 Storage Data Platform
2 Product Information
3 Product Usage Instructions
4 Abstract
5 Migration Options
6 Overview – Migration without Downtime
7 Steps:
8 Monitoring Migration
9 Performance Optimizing S3A for Migration
10 Multipart Size
11 Documents / Resources
11.1 References
12 Related Posts

VAST S3 Storage Data Platform

Product Information

Specifications

- Product Name: VAST S3 Migration Tool
- Version: 1.0
- Target Audience: Data engineers, data architects, system administrators
- Compatibility: Hadoop, Hive, S3 storage configurations

Product Usage Instructions

Option 1: Migrating Data to S3 with DistCp using the S3A Adapter

- **Use Case:** Ideal for bulk data transfer scenarios.
- **Purpose:** Efficiently move large amounts of raw data from HDFS to S3.

Steps:

1. Configure /etc/hadoop/conf/core-site.xml:

```
fs.s3a.access.key RBY76D9A..
fs.s3a.secret.key aBa6Odt/u/ua2J3ACABpa2..
fs.s3a.path.style.access true
fs.s3.enable-storage-classes true
fs.s3a.connection.ssl.enabled false
fs.s3a.endpoint http://vippool.yourvastcluster.com
```

Make sure the fs.s3a.endpoint is a DNS name for the VIP-pool.

2. Execute DistCp Command:
   hadoop distcp hdfs://your-hdfs-path/target_data/ s3a://vast-hadoop-s3-bucket/target_data
3. Verify Data Transfer:
   hdfs dfs -ls s3a://vast-hadoop-s3-bucket/target_data

Option 2: Migrating Hive Tables and Using Hive-Meta with S3A

- **Use Case:** Ideal for preserving Hive table metadata.
- **Purpose:** Migrate Hive table structures, schema, and metadata from HDFS to S3.

Frequently Asked Questions (FAQ)

- **Q: Who is the target audience for this tool?**
  A: Data engineers, data architects, and system administrators managing data storage and migration within Hadoop environments.
- **Q: What are the prerequisites for using this migration tool?**
  A: Familiarity with Hadoop, Hive, and S3 storage configurations is recommended.

Best Practices Guide for Migrating HDFS Data & Hive Tables to VAST S3

Abstract

This guide provides a comprehensive overview of best practices for migrating data from traditional Hive/HDFS tables to VAST S3 storage using the S3A adapter. It offers multiple options tailored to different use cases, including bulk data transfer, preserving Hive table metadata, ensuring data consistency, and minimizing downtime. By following the steps outlined in the relevant sections, data engineers and administrators can choose the most suitable migration method based on their specific requirements, ensuring efficient data transfer with minimal impact on ongoing operations. This guide includes prerequisites, detailed migration steps, and optimization tips to facilitate a smooth and effective migration process.

Audience
This guide is intended for data engineers, data architects, and system administrators responsible for managing data storage and migration within Hadoop environments. Familiarity with Hadoop, Hive, and S3 storage configurations is recommended.

Prerequisites

- VAST user with the appropriate permissions & S3 access, secret keys: Managing Users
- S3 Bucket on VAST cluster: Managing Buckets
- Assigned Identity policy for the bucket-owner: Managing S3 Access

- Network connectivity between VAST VIPs and the Hadoop cluster: Network Configuration
- Hadoop installed on-premises with DistCp

Migration Options

Option 1: Migrating Data to S3 with DistCp using the S3A Adapter

Use Case
This option is ideal for bulk data transfer scenarios where the primary goal is to move large amounts of raw data from HDFS to S3 efficiently. It is suitable when preserving Hive table metadata is not a concern.

Purpose
The main purpose of using DistCp (Distributed Copy) with the S3A adapter is to leverage its parallel copying capabilities to quickly and effectively transfer data from HDFS to S3 storage. This method ensures that the data is moved in a scalable and robust manner.

Steps

1. Configure /etc/hadoop/conf/core-site.xml: Update the Hadoop configuration file to include S3A credentials and endpoint information.
    - <property>
    - <name>fs.s3a.access.key</name>
    - <value>RBY76D9A..</value>
    - </property>
    - <property>
    - <name>fs.s3a.secret.key</name>
    - <value>aBa6Odt/u/ua2J3ACABpa2..</value>
    - </property>
    - <property>
    - <name>fs.s3a.path.style.access</name>
    - <value>true</value>
    - </property>
    - <property>
    - <name>fs.s3.enable-storage-classes</name>
    - <value>true</value>
    - </property>
    - <property>
    - <name>fs.s3a.connection.ssl.enabled</name>
    - <value>false</value>
    - </property>
    - <property>
    - <name>fs.s3a.endpoint</name>
    - <value>http://vippool.yourvastcluster.com</value>
    - </property>
    - Make sure the fs.s3a.endpoint is a DNS name for the VIP-pool, so the balancing will be activated between the CNODE’s.

2. Execute DistCp Command: Use the DistCp command to copy data from the HDFS path to the target S3 bucket.
   hadoop distcp hdfs://your-hdfs-path/target_data/ s3a://vast-hadoop-s3-bucket/target_data
3. Verify Data Transfer: After the transfer, verify that the data has been copied correctly by listing the contents of the S3 bucket.
   hdfs dfs -ls s3a://vast-hadoop-s3-bucket/target_data

Option 2: Migrating Hive Tables and Using Hive-Meta with S3A

Use Case:
This option is ideal for scenarios where it is crucial to migrate Hive tables along with their metadata to ensure that the Hive schema and table definitions are preserved. It is suitable for maintaining the integrity and functionality of Hive tables after migration to S3.

Purpose:
The primary purpose of this method is to seamlessly migrate Hive table structures, including schema definitions and metadata, from HDFS to S3. This ensures that the Hive tables remain queryable and maintain their defined structure in the new storage environment.

Hive S3A Configuration
Steps:
Replace <bucket-name> with the name of your S3A bucket. This example creates a Hive database named mydb and a table named mytable stored as a text file in S3A.

1. Configure Hive for S3A: Set the S3A connector as the default filesystem for Hive. set fs.defaultFS=s3a://<bucket-name>/
2. Create Hive Database: Create a new Hive database that will store the migrated tables. CREATE DATABASE mydb;
3. Create Hive Table in S3: Create a table in the Hive database that references data stored in S3. CREATE TABLE mydb.mytable (col1 INT, col2 STRING)
   STORED AS TEXTFILE
   LOCATION ‘s3a://<bucket-name>/mytable/’;
4. Verify Table Creation: Ensure that the table has been created successfully and is pointing to the correct S3 location.
   SHOW TABLES;

By following these steps, you can effectively migrate Hive tables to S3, preserving the schema and metadata, and ensuring that the tables remain functional and queryable in their new location. This method is essential for scenarios where maintaining the logical structure of the Hive tables is critical.

Overview – Migration without Downtime

Use Case:
This method is ideal for scenarios where the data needs to be migrated to a new storage location without causing any downtime or disruption to ongoing operations.

Purpose:
To migrate data from one storage location to another seamlessly, ensuring continuous availability of the data during the migration process.

Steps

1. Create a New Table: Create a new table in the desired storage location using the same schema as the original table. CREATE TABLE newtable LIKE mytable;
2. Copy Data to the New Table: Use the INSERT INTO statement to copy the data from the original table to the new table.
   INSERT INTO newtable SELECT * FROM mytable;
3. Update References to the Original Table: Once the data has been copied, update any references to the original table to point to the new table.
   ALTER TABLE mytable_external SET LOCATION 'new_location' ;
4. Drop the Original Table: After ensuring that all references have been updated and the new table is functioning correctly, drop the original table to free up resources.
   DROP TABLE mytable;

By following these steps, you can migrate your Apache Hive table from one storage location to another without any downtime, ensuring continuous data availability and minimal disruption to your operations.

Option 2a: Migration using Hive Snapshots (CTAS)

Use Case:
This method is ideal for scenarios requiring a transactionally consistent snapshot of the data. It is useful for data migration, backups, or analytics on a stable snapshot.

Purpose:
The primary purpose of CTAS (Create Table As Select) is to create a consistent snapshot of the data at a specific point in time. This ensures that all changes to the data up until the point of the snapshot are included, providing a stable data state for migration or analysis.

Steps:

1. Create Snapshot (Source Table): Create a snapshot of the source table using the CREATE TABLE AS SELECT statement. This command creates a new table store _sales_snap and populates it with data from store _ sales. This ensures any changes made to the store _ sales table after this point do not affect the migration process.
   CREATE TABLE store_sales_snap AS SELECT * FROM store_sales;
2. Export Snapshot (Source Table): Export the snapshot data to the desired S3 location using the INSERT OVERWRITE DIRECTORY command. This moves the data from HDFS to the VAST S3 bucket, which is the destination storage. INSERT OVERWRITE DIRECTORY 's3://my-s3-bucket/export -path' SELECT * FROM store_sales_snap;
3. Restore Table (Destination Table): Create a new table in the destination and point it to the exported data on S3. The CREATE TABLE … LIKE statement creates a new table destination _ store _ sales with the same schema as store _ sales. The ALTER TABLE … SET LOCATION statement changes the location of destination _ store _ sales to the S3 path where the snapshot data was exported.
   CREATE TABLE destination_store_sales LIKE store_sales; ALTER TABLE destination_store_sales SET LOCATION 's3://my-s3-bucket/export-path';

In this migration process, a snapshot of the source table (store _ sales) is created and exported to an S3 bucket. A new table (destination _ store _ sales) is then created at the destination with the same schema and is linked to the exported data on S3. This method ensures a consistent and isolated migration of the data from the source to the destination.

Option 2b: Migration Using S3 Temp Bucket

Use Case:
This method is ideal for scenarios where you need to ensure data consistency during migration by using a temporary S3 bucket as an intermediary storage location.

Purpose:
The primary purpose of using a temporary S3 bucket is to provide a staging area that ensures data consistency during the migration process from HDFS to VAST S3.

Steps:

1. Export the Source Table to a Temporary S3 Bucket: Copy data from the store _ sales table on HDFS to a temporary location on S3. This is done using the EXPORT TABLE statement.
   EXPORT TABLE store_sales TO ‘s3://your_temp_bucket/store_sales_temp’;
2. Create the Target Table in Hive: Define the schema and location for the target table on S3. Use the CREATE EXTERNAL TABLE statement to create a table schema similar to the store _ sales table and specify the data storage format (e.g., PARQUET).
   CREATE EXTERNAL TABLE store_sales_s3 (
   - ss_sold_date_sk INT,
   - ss_sold_time_sk INT,
   - ss_item_sk INT,
   - ss_customer_sk INT,
   - ss_cdemo_sk INT,
   - ss_hdemo_sk INT,
   - ss_addr_sk INT,
   - ss_store_sk INT,
   - ss_promo_sk INT,
   - ss_ticket_number INT,
   - ss_quantity INT,
   - ss_wholesale_cost DECIMAL(7,2),
   - ss_list_price DECIMAL(7,2),
   - ss_sales_price DECIMAL(7,2),
   - ss_ext_discount_amt DECIMAL(7,2),
   - ss_ext_sales_price DECIMAL(7,2),
   - ss_ext_wholesale_cost DECIMAL(7,2),
   - ss_ext_list_price DECIMAL(7,2),
   - ss_ext_tax DECIMAL(7,2),
   - ss_coupon_amt DECIMAL(7,2),
   - ss_net_paid DECIMAL(7,2),
   - ss_net_paid_inc_tax DECIMAL(7,2),
   - ss_net_profit DECIMAL(7,2)
   - STORED AS PARQUET

- LOCATION ‘s3://your_target_bucket/store_sales_s3’;

3. Import the Data from the Temporary S3 Bucket to the Target Table: Populate the store _sales_s3 table with data from the temporary S3 bucket. Use the INSERT OVERWRITE TABLE statement to copy data from the temporary S3 location to the store _sales_s3 table on S3.
   INSERT OVERWRITE TABLE store_sales_s3 SELECT * FROM ‘s3://your_temp_bucket/store_sales_temp’;
4. Drop the Temporary S3 Bucket and Its Contents: Clean up by deleting the temporary data. After data migration is complete, temporary storage is no longer needed. Use the Hadoop file system command to remove the temporary S3 bucket. hadoop fs -rm -r s3a://your_temp_bucket/store_sales_temp

This method facilitates efficient data migration from HDFS to S3 using a temporary S3 bucket as intermediary storage. It ensures data consistency and allows for schema and storage format definitions in the target location.

Option 2c: Migrating Tables Data Using Simple INSERT Statements

Use Case
This method is ideal for straightforward migrations where data needs to be copied from a source table on HDFS to a target table on VAST S3 without the need for intermediate steps or complex configurations.

Purpose:
The primary purpose is to create a new table on VAST S3 and copy the data from the source table on HDFS directly using simple Hive SQL statements.

Steps:

1. Create the Target Table on S3: Create a new table on VAST S3 with the same schema as the source table on HDFS.
   Use the CREATE EXTERNAL TABLE statement to define the table schema and specify the data storage format (e.g., PARQUET) and location on S3.
   CREATE EXTERNAL TABLE store_sales_s3 (
   - ss_sold_date_sk INT,
   - ss_sold_time_sk INT,
   - ss_item_sk INT,
   - ss_customer_sk INT,
   - ss_cdemo_sk INT,
   - ss_hdemo_sk INT,
   - ss_addr_sk INT,
   - ss_store_sk INT,
   - ss_promo_sk INT,
   - ss_ticket_number INT,
   - ss_quantity INT
   -)
   - STORED AS PARQUET
   - LOCATION ‘s3://your_target_bucket/store_sales_s3’;
2. Copy Data from the Source Table to the Target Table: Use the INSERT INTO statement to copy data from the source table on HDFS to the target table on S3.

INSERT INTO store_sales_s3 SELECT * FROM store_sales;

3. Validate Data Migration: Ensure that the data has been successfully written to the target table on S3. Use a SELECT COUNT(*) query to retrieve the number of rows in the target table and compare it with the source table to validate that all records have been migrated.ds

By using simple CREATE TABLE, INSERT INTO, and SELECT COUNT(*) statements, you can effectively migrate data from a source table on HDFS to a target table on VAST S3. This method ensures that the schema is maintained and
allows for straightforward validation of the data migration.

Option 2d: Migrating External Table to VAST S3 from Hive on HDFS Table

Use Case:
This method is ideal for migrating external tables from Hive on HDFS to VAST S3, maintaining the logical structure and partitioning of the dataset for optimized query performance.

Purpose:
The primary purpose is to create a new partitioned table on VAST S3 with a schema matching the source table from HDFS. This ensures that metadata is stored in Hive while the actual data resides on S3, allowing for efficient data storage and retrieval.

Steps:

1. Create the Target S3 Table: Create a new partitioned external table on VAST S3 with a schema that matches the source table on HDFS. Use the CREATE EXTERNAL TABLE statement to define the table schema, specify the data format (e.g., PARQUET), and set the location to a VAST S3 bucket.
   CREATE EXTERNAL TABLE tlc_taxi_data_s3_partitioned (
   VendorID INT,
   - tpep_pickup_datetime TIMESTAMP,
   - tpep_dropoff_datetime TIMESTAMP,
   - passenger_count BIGINT,
   - trip_distance DOUBLE,
   - RatecodeID BIGINT,
   - store_and_fwd_flag STRING,
   - PULocationID INT,
   - DOLocationID INT,
   - payment_type BIGINT,
   - fare_amount DOUBLE,
   - extra DOUBLE,
   - mta_tax DOUBLE,
   - tip_amount DOUBLE,
   - tolls_amount DOUBLE,
   - improvement_surcharge DOUBLE,
   - total_amount DOUBLE,
   - congestion_surcharge DOUBLE,
   - Airport_fee DOUBLE
   -)

- PARTITIONED BY (year STRING, month STRING)
- STORED AS PARQUET
- LOCATION ‘s3a://cloudera/hive/tlc_taxi_data_s3_partitioned’
- TBLPROPERTIES (‘external.table.purge’=’true’);
- Partitioning: The PARTITIONED BY clause specifies that the data should be partitioned by year and month, which optimizes query performance.
- Location: The LOCATION specifies the VAST S3 path where the data will be stored.
- Table Properties: TBLPROPERTIES is set to ensure that when the table is dropped, the data remains in S3.

2. Populate the Partitioned Table Using the Hive on HDFS Table: Use the INSERT INTO TABLE statement to populate the tlc _ taxi _ data _ s3 _ partitioned table with data from the source table.
    - The PARTITION clause ensures that the data is partitioned by year and month while being written to S3.
    - INSERT INTO TABLE tlc_taxi_data_s3_partitioned PARTITION (year, month) SELECT
    - VendorID,
    - tpep_pickup_datetime,
    - tpep_dropoff_datetime,
    - passenger_count,
    - trip_distance,
    - RatecodeID,
    - store_and_fwd_flag,
    - PULocationID,
    - DOLocationID,
    - payment_type,
    - fare_amount,
    - extra,
    - mta_tax,
    - tip_amount,
    - tolls_amount,
    - improvement_surcharge,
    - total_amount,
    - congestion_surcharge,
    - Airport_fee,
    - SUBSTRING(INPUT__FILE__NAME, -16, 4) AS year,
    - SUBSTRING(INPUT__FILE__NAME, -11, 2) AS month
    - FROM tlc_taxi_intermediary;

SUBSTRING Functions: Extract the year and month information from the file name, assuming a specific naming convention.
By creating an external partitioned table on VAST S3 and populating it with data from the Hive table on HDFS, this method ensures efficient data storage and retrieval while maintaining the logical structure of the dataset. This approach leverages partitioning to optimize query performance and provides a seamless migration path for external tables.

Monitoring Migration

To monitor the progress of the export and import process in Apache Hive, various tools and techniques can be used. Here are several options to consider:

1. Hive CLI or Beeline:
   - Use the Hive command-line interface (CLI) or Beeline to monitor the progress of export and import operations. When you execute the queries, the CLI or Beeline displays the query progress and status.
   - Commands: You can monitor the progress by checking the logs or using the SHOW JOBS or SHOW SESSIONS commands to view the status of the running jobs or sessions.
2. Hadoop Resource Manager:
   - The Hadoop Resource Manager provides a web interface to monitor the progress of Hive export and import operations.
   - Features: View the status of running jobs and tasks, check resource usage, and monitor task progress through the Resource Manager's web interface.
3. Third-Party Monitoring Tools:
   - Utilize third-party monitoring tools such as Ganglia, Nagios, or Datadog to monitor the Hive service.
   - Benefits: These tools offer various metrics and visualizations to help you track the performance and status of the Hive service and its components. They provide additional insights into system performance and can alert you to any issues that may arise during the migration process.

By employing these tools and techniques, you can effectively monitor the progress of your data migration, ensuring that export and import operations are proceeding as expected and allowing you to address any issues that may arise promptly.

Performance Optimizing S3A for Migration

S3A is a file system implementation for Apache Hadoop that allows Hadoop applications to read and write data to S3 storage. It provides an alternative to Hadoop's native HDFS file system, enabling users to store and access data on S3 efficiently. Optimizing S3A settings can significantly improve the performance of data migration from HDFS to VAST S3.

Basic Configuration (No Optimizations):
Below is an example of a basic core-site.xml configuration for S3A without any tuning:

- `<configuration>`
- `<property>`
- `<name>fs.defaultFS</name>`
- `<value>s3a://temp1</value>`
- `</property>`
- `<property>`
- `<name>hadoop.tmp.dir</name>`
- `<value>/home/hadoop/tmp</value>`
- `</property>`
- `<property>`
- `<name>fs.s3a.access.key</name>`
- `<value>AG8SSUT6SE436AEXBPRE</value>`
- `</property>`

- <property>
- <name>fs.s3a.secret.key</name>
- <value>SIOPRO3jsvT1maTyMxetaOvXDpRsyrAX78zcEVEEE</value>
- </property>
- <property>
- <name>fs.s3a.path.style.access</name>
- <value>true</value>
- </property>
- <property>
- <name>fs.s3.enable-storage-classes</name>
- <value>true</value>
- </property>
- <property>
- <name>fs.s3a.connection.ssl.enabled</name>
- <value>false</value>
- </property>
- <property>
- <name>fs.s3a.endpoint</name>
- <value>http://vippool.yourvastcluster.com</value>
- </property>
- </configuration>

TTL Zero Tuning:
The S3A client caches the endpoint connection for performance reasons. To improve performance, set the TTL (time to live) to zero, ensuring VAST's ability to scale over multiple Cnodes.
Add the following to core-site.xml:

- <property>
- <name>fs.s3a.endpoint.connection.ttl</name>
- <value>0</value>
- </property>

Additional Tuning for S3A:
Additional parameters can be tuned to optimize S3A performance:

Multipart Size

- Purpose: Specifies the size of each part when uploading large files to S3. Increasing this size can improve upload performance for larger files by reducing the number of parts.
- Default Value: 128 MB
- Optimal Value: The optimal value depends on network bandwidth, the size of the data being uploaded, and the S3 storage class. Larger values improve performance for large files but may increase the risk of failed uploads due to network issues. For instance, setting the multipart size to 10 MB can enhance the upload speed for smaller files but may not be optimal for very large files, which might benefit from a larger multipart size.
    - <property>

    - <name>fs.s3a.multipart.size</name> <value>10M</value>
    - </property>

Fast Upload Active Blocks

- Purpose: Determines the maximum number of active blocks to upload in parallel during a fast upload. This can significantly improve performance for large files.
- Default Value: 4
- Optimal Value: The optimal value depends on network bandwidth, the number of available cores, and the S3 storage class. Higher values allow more parallel uploads but can increase network bandwidth usage and connections to S3. For example, setting it to 100 can enhance upload throughput if the network bandwidth and S3 service can handle the increased number of connections.
    - <property>
    - <name>fs.s3a.fast.upload.active.blocks</name> <value>100</value>
    - </property>

Maximum Threads

- Purpose: Specifies the maximum number of threads that can be used by the S3A filesystem connector for parallel operations. This includes uploading, downloading, listing objects, and deleting objects.
- Default Value: 256
- Optimal Value: The optimal value depends on network bandwidth, S3 storage class, and available client/server resources. Increasing the value can improve parallel operations but may increase resource usage and latency. For example, if the system's network bandwidth and CPU resources can handle it, increasing the number of threads to 100 can improve the speed of concurrent operations.
    - <property>
    - <name>fs.s3a.threads.max</name> <value>100</value>
    - </property>

Block Size

- Purpose: Sets the block size for a file stored in S3. Files are divided into blocks, and each block is stored as a separate S3 object.
- Default Value: 32 MB
- Optimal Value: The optimal value depends on file size, access patterns, and network bandwidth. Larger block sizes reduce the number of S3 objects created and improve read/write performance for large files. Smaller block sizes are more appropriate for small files or infrequently accessed data. For instance, setting the block size to 100 MB can be beneficial for workloads involving large, sequentially accessed files.
    - <property>
    - <name>fs.s3a.block.size</name> <value>100m</value>
    - </property>

Additional Tuning for VAST:
From VAST version 4.7 SP10 onwards, there is an optimization setting (vtool) for handling Hadoop-based data to

VAST S3. This setting can significantly enhance the performance of data migrations.

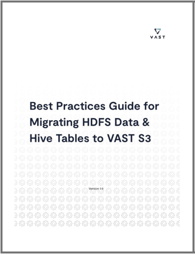
Steps:

1. Configure HDFS Cluster:
   Ensure that multipart uploads are disabled by setting the threshold and size to exceed the actual size of your files. For example, if your files are generally 1 GB or larger, set the threshold and size to 1 GB. Adjust accordingly if your files are larger. Disabling multipart uploads for smaller files simplifies the upload process and reduces overhead.
   - Configure core-site.xml: <property>
   - …
   - <name>fs.s3a.multipart.threshold</name> <v alue>1G</value>
   - </property>
   - <property>
   - <name>fs.s3a.multipart.size</name> <value>1G</value>
   - </property>
   - <property>
   - <name>fs.s3a.fast.upload</name> <value>true</value>
   - …
   - </property>
2. Apply VAST Optimization:
   - Use SSH to connect to one of the CNODEs on VAST.
   - Apply the optimization setting using the vtool command. This setting optimizes the copy process by using links, reducing the time and resources required for data migration.
     vtool vsettings set S3_COPY_USING_LINK=true

By implementing these configurations and optimizations, you can significantly improve the performance of S3A for data migration from HDFS to VAST S3, ensuring a more efficient and scalable data transfer process.

For more information on Universal Storage and how it can help you solve your application problems, reach out to us at hello@vastdata.com.

Documents / Resources

 Best Practices Guide for Migrating HDFS Data & Hive Tables to VAST S3	VAST S3 Storage Data Platform [pdf] User Guide S3, S3 Storage Data Platform, Storage Data Platform, Data Platform, Platform

References

- **User Manual**