This article describes considerations and processes for migrating the data from an Apache HBase cluster to a Cloud Bigtable cluster on Google Cloud (Google Cloud).

Before you begin this migration, you should consider performance implications, Bigtable schema design, implications for your approach to authentication and authorization, and the Bigtable feature set.

Under a typical workload, Bigtable delivers highly predictable performance. When everything is running smoothly, you can expect to achieve the following performance for each node in your Bigtable cluster, depending on which type of storage your cluster uses.

Storage Type	Reads		Writes	Scans
SSD	10,000 rows per second @ 6 ms	**or**	10,000 rows per second @ 6 ms	220 MB/s
HDD	500 rows per second @ 200 ms	**or**	10,000 rows per second @ 50 ms	180 MB/s

The estimates shown in the list are based on rows that contain 1 KB of data. They also reflect a read-only or write-only workload. Performance for a mixed workload of reads and writes will vary.

These performance numbers are guidelines, not hard and fast rules. Per-node performance may vary based on your workload and the typical value size of a request or response. For more information, see Understanding Bigtable Performance (/bigtable/docs/performance).

Designing a Bigtable schema is not like designing a schema for a relational database. Before designing your schema, review the concepts laid out in Designing Your Schema (/bigtable/docs/schema-design).

You must also keep the schema below the recommended limits for size (/bigtable/quotas#limits). As a guideline, keep single rows below 100 MB and single values below 10 MB. There may be scenarios in which you'll need to store large values. Storing large values can impact performance, because

extracting large values requires time as well as memory. Evaluate those scenarios on a case-by-case basis.

Before you design access control for Bigtable, review the existing HBase authentication and authorization processes.

Bigtable uses Google Cloud's standard mechanisms for authentication (/docs/authentication/) and Cloud Identity and Access Management to provide access control, so you convert your existing authorization on HBase to Cloud IAM. You can map the existing Hadoop groups that provide access control mechanisms for HBase to different service accounts.

While Bigtable allows you to control access at the instance level, it does not provide fine-grained control at the table level. One way to provide table-level granularity is to group tables that have similar access patterns under a Bigtable instance. But this approach could mean that you will need to use multiple Bigtable instances to migrate all your tables.

For more information, see Access Control (/bigtable/docs/access-control).

To migrate your data from HBase to Bigtable, you export the data as a series of Hadoop sequence files. This is a file format used by HBase consisting of binary key/value pairs.

To migrate the HBase table to Bigtable, follow these steps:

1. Collect details from HBase.
2. Export HBase tables to sequence files.
3. Move the sequence files to Cloud Storage.
4. Import the sequence files into Bigtable using Dataflow.
5. Validate the move.

To prepare for the migration, gather the following information from the existing HBase cluster, because you will need this information to build the destination table.

- List of tables
- Row counts
- Cell counts
- Column family details (including Time to Live, maximum number of versions)

An easy way to collect these details on a source table is to use the following script, which leaves the result on HDFS:

When you know the basics of the HBase tables to be migrated, you need to export the table to sequence files and move them to Cloud Storage. Before taking actions to migrate any online data, run the following steps to make sure that your cluster meets the prerequisites for accessing Google Cloud:

- Install the Cloud Storage connector

  If you want to migrate online data using `distcp`, you must install and configure the Cloud Storage connector. First, identify the HDFS file system that manages the data you want to migrate. Next, determine which client node in your Hadoop cluster has access to this file system. Finally, install the connector on the client node. For detailed installation steps, see Installing the Cloud Storage connector (/dataproc/docs/concepts/connectors/install-storage-connector).

- Install the Cloud SDK

  To migrate data with either `distcp` or `gsutil`, install the Cloud SDK on the Hadoop cluster client node where the migration will be initiated. For detailed installation steps, see the Cloud SDK documentation (/sdk/docs/).

Next, export the HBase table you want to migrate to somewhere in your Hadoop cluster. Let's assume your HBase table name is [MY_NEW_TABLE]. The target directory is under your user directory in HDFS. Export the HBase table as sequence files using the following commands:

The next step is to move the sequence files to a Cloud Storage bucket. Depending upon the size of the data, the number of files, source of the data, and the available bandwidth, you can choose the appropriate option to move sequence files to Cloud Storage: Transfer Appliance, `distcp`, `gsutil`, or Storage Transfer Service.

Use Transfer Appliance to migrate your data when:

- You want to control the outgoing bandwidth on a schedule.

- The size of the data is greater than 20 TB.

With this option, you do not need to buy or provision an extra network with Google. The end-to-end transfer time (appliance shipment time, rehydration, etc.) averages 100 Mbps.

To move data with Transfer Appliance, copy the sequence files to it and then ship the appliance back to Google. Google loads the data onto Google Cloud. For more information, see this Transfer Appliance documentation (/transfer-appliance/docs/2.0/).

Use `distcp` to migrate your data when:

- More than 100 Mbps bandwidth is available for the migration.
- The Cloud Storage connector and the Cloud SDK can be installed on the source Hadoop environment.
- Managing a new Hadoop job to perform the data migration is acceptable.
- The size of the data is less than 20 TB.

To move data with `distcp`, use your Hadoop cluster client node configured with the Cloud Storage connector to submit a MapReduce job to copy the sequence files to Cloud Storage:

Use `gsutil` to migrate your data when:

- More than 100 Mbps bandwidth is available for the migration.
- The Cloud SDK can be installed on the source Hadoop environment.
- Managing a new Hadoop job to perform the data migration is not acceptable.
- The size of the data is less than 10 TB.

To move data with `gsutil`, use your Hadoop cluster client node to initiate the data migration:

Use Storage Transfer Service to migrate your data when:

- Your data source is an Amazon S3 bucket, an HTTP/HTTPS location, or a Cloud Storage bucket.
- The size of the data is less than 10 TB.

Storage Transfer Service has options that make data transfers and synchronization between data sources and data sinks easier. For example, you can:

- Schedule one-time transfer operations or recurring transfer operations.
- Delete existing objects in the destination bucket if they don't have a corresponding object in the source.
- Delete source objects after transferring them.
- Schedule periodic synchronization from data source to data sink with advanced filters based on file creation dates, file-name filters, and the times of day you prefer to import data.

For more information, see this Storage Transfer Service (/storage/transfer/) documentation.

The next step is to create the destination table in Bigtable.

First, you use the `gcloud` command-line tool to install the Bigtable client tool `cbt`.

Next, you create a table in Bigtable with the appropriate column families from your previous discovery effort.

Using the existing splits, presplit the destination table as you create it. This will improve the bulk load performance.

You will need to create the table with predefined splits just before you initiate the large load. If there is a gap between ıe table is created and time of the actual load, the empty table with splits may lose those splits. That's because Bigta e any empty splits it finds during the next compaction. If you create splits to improve the initial bulk load, then be sur esplitting immediately before the kickoff of that bulk load.

For example, if you find that the existing splits are:

```
'15861', '29374', '38173', '180922', '203294', '335846', '641111', '746477', '807307',
'871053', '931689', '1729462', '1952670', '4356485', '4943705', '5968738', '6917370',
'8993145', '10624362', '11309714', '12056747', '12772074', '14370672', '16583264',
'18835454', '21194008', '22021148', '23702800', '25532516', '55555555'
```

Then set up a default project and a Bigtable instance for the `cbt` tool for your user account like this:

Create these splits in the destination table:

Create column families in the destination table to match the ones you discovered earlier. For example, if you discovered that there are two column families, `cf1` and `cf2`, create the column family `cf1` on Bigtable like this:

Create the column family `cf2` like this:

After creating the column families, it is important to update each column family's garbage collection policy, including the maximum age and maximum number of versions for values in that column family. You must do this even if you used HBase's default settings for your HBase table, because Bigtable's native tools use a different default setting than HBase.

There are two ways to import data to Bigtable. See Importing Sequence Files (/bigtable/docs/importing-sequence-files#import-table) in the Bigtable docs for details.

Keep the following tips in mind:

- To improve the performance of data loading, be sure to set `maxNumWorkers`. This value helps to ensure that the import job has enough compute power to complete in a reasonable amount of time, but not so much that it would overwhelm the Bigtable cluster.
- During the import, you should monitor the Bigtable cluster's CPU usage. The CPU usage section (/bigtable/docs/monitoring-instance#cpu) of the Bigtable monitoring document provides more information about this topic. If the CPU utilization across Bigtable cluster is too high, you might need to add additional nodes. It may take up to 20 minutes for the cluster to provide the performance benefit of additional nodes.

For more information about monitoring the Bigtable instance, see Monitoring a Bigtable Instance (/bigtable/docs/monitoring-instance).

To validate the imported data, you can use a couple of different checks:

- **Checking the row count match.** The Dataflow job will report the total row count. This value needs to match the source HBase table row count.
- **Spot-checking with specific row queries.** You can pick up a specific set of row keys from the source table and query them on the destination table to make sure they match:

- Check out the other parts of the Hadoop migration guide:
  - Overview (/solutions/migration/hadoop/hadoop-gcp-migration-overview)
  - Data migration guide (/solutions/migration/hadoop/hadoop-gcp-migration-data)
  - Job migration guide (/solutions/migration/hadoop/hadoop-gcp-migration-jobs)
- Learn more about Cloud Storage (/storage/docs/).