This guide describes the process of moving data from on-premises Hadoop Distributed File System (HDFS) to Google Cloud (Google Cloud).

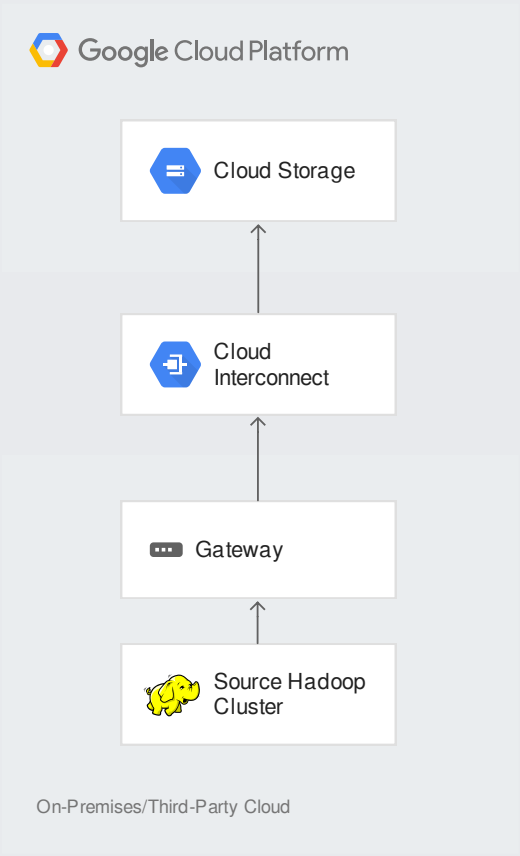
This is the second of four guides describing how to move from on-premises Hadoop:

- Migrating On-Premises Hadoop Infrastructure to Google Cloud (/solutions/migration/hadoop/hadoop-gcp-migration-overview) provides an overview of the migration process, with particular emphasis on moving from large, persistent clusters to an ephemeral model.
- This guide, focused on moving your data to Google Cloud.
- Migrating data from HBase to Cloud Bigtable (/solutions/migration/hadoop/hadoop-gcp-migration-data-hbase-to-bigtable)
- Migrating Hadoop Jobs from On-Premises to Dataproc (/solutions/migration/hadoop/hadoop-gcp-migration-jobs) describes the process of running your jobs on Dataproc and other Google Cloud products.

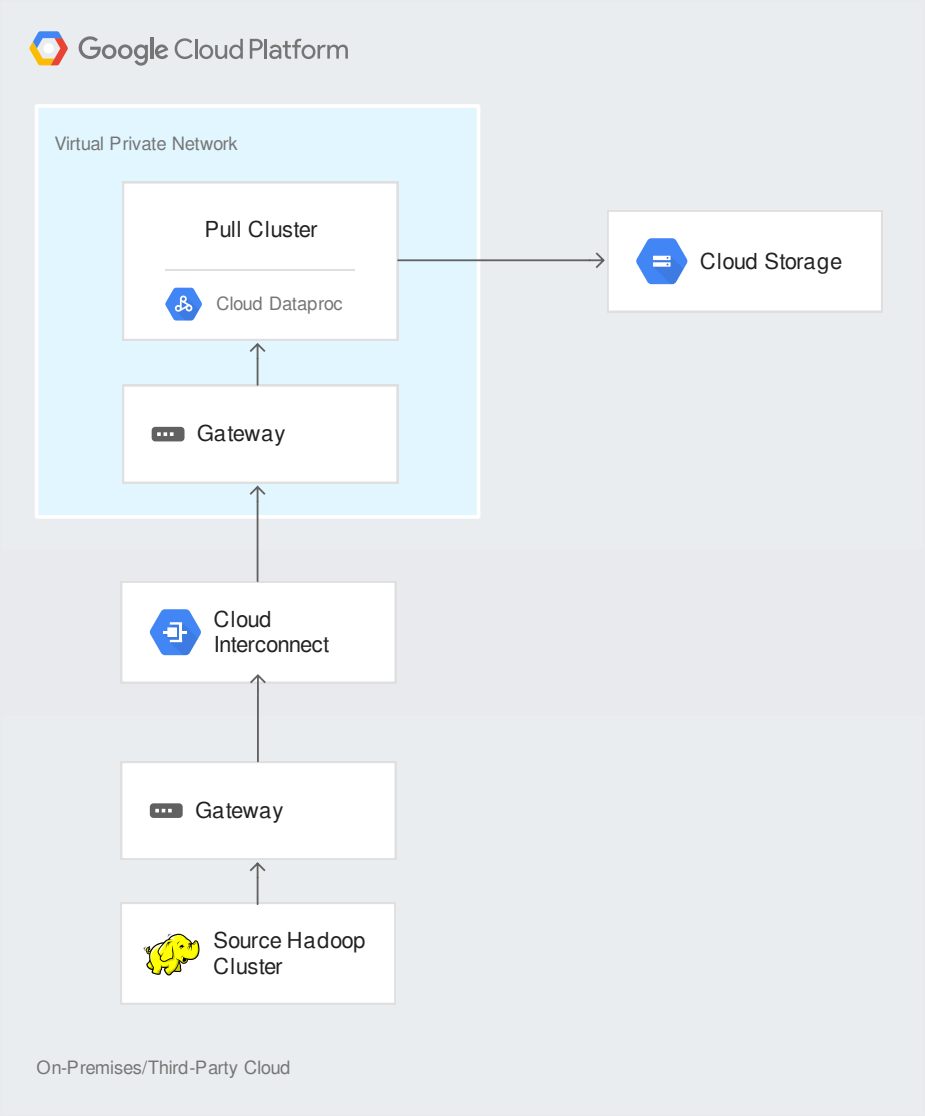
The following sections describe best practices for planning your data migration from on-premises HDFS to Google Cloud. Plan to migrate incrementally so you can leave time to migrate jobs,experiment, and test after moving each body of data.

There are two different migration models you should consider for transferring HDFS data to the cloud: push and pull. Both models use Hadoop DistCp (https://hadoop.apache.org/docs/current/hadoop-distcp/DistCp.html) to copy data from your on-premises HDFS clusters to Cloud Storage, but they use different approaches.

The push model is the simplest model: the source cluster runs the `distcp` jobs on its data nodes and pushes files directly to Cloud Storage, as shown in the following diagram:



The pull model is more complex, but has some 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, as shown in the following diagram:



The push model is the simplest model because the source cluster can push data directly to Cloud Storage and you don't need to create extra compute resources to perform the copy. However, if you intend to continue using the source cluster during the migration for other regular data processing jobs, you should ensure that enough resources, such as CPU, RAM, and network bandwidth, are available on the source cluster to also perform the copy jobs.

If the source cluster is already running at compute capacity, and if you cannot increase the resources on the source cluster to perform the copy, then you should consider using the pull model instead.

While more complex than the push model, the pull model has a number of advantages:

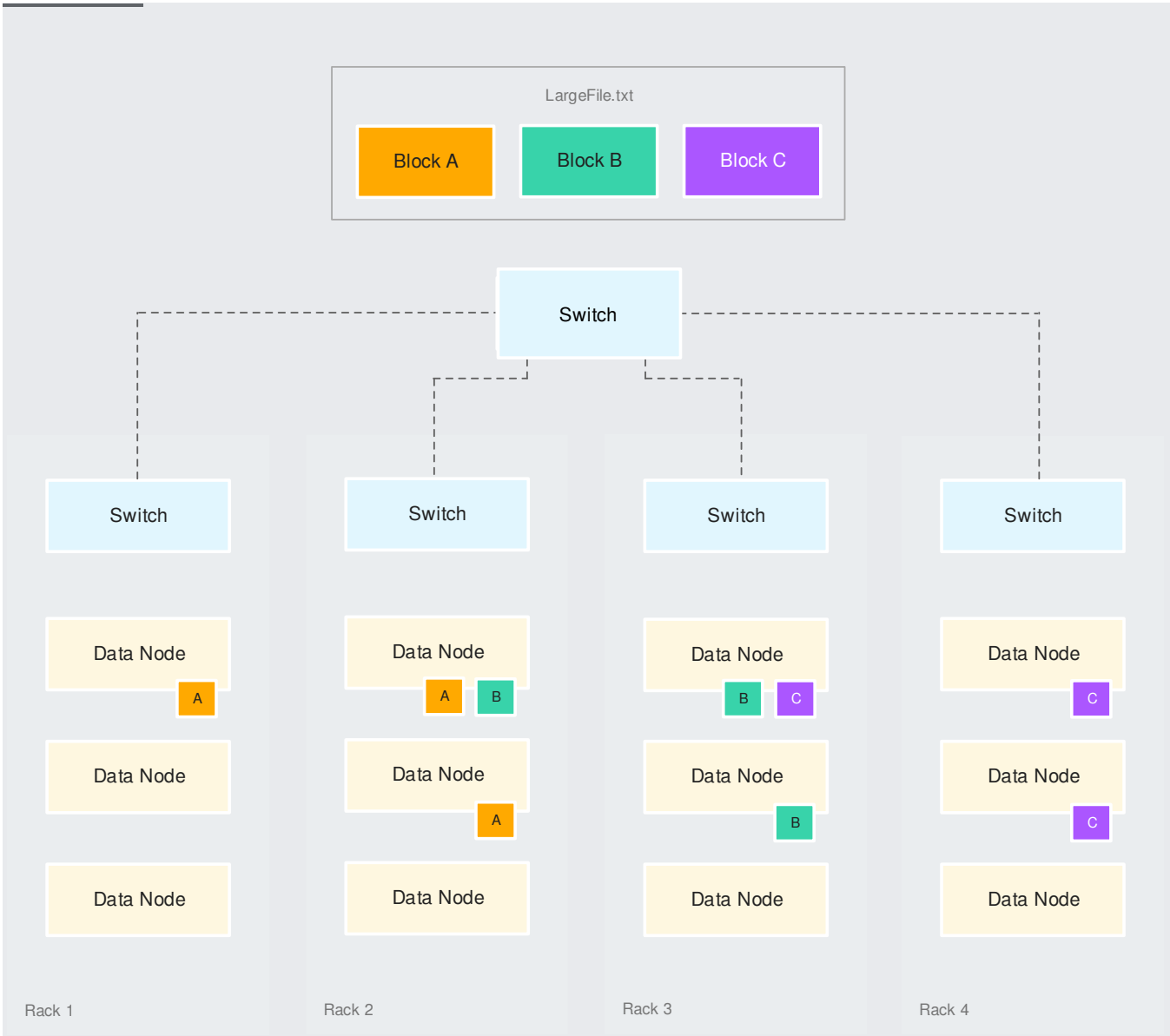
- Impact on the source cluster's CPU and RAM resources is minimized, because the source nodes are used only for serving blocks out of the cluster. You can also fine-tune the specifications of

the pull cluster's resources on Google Cloud to handle the copy jobs, and tear down the pull cluster when the migration is complete.

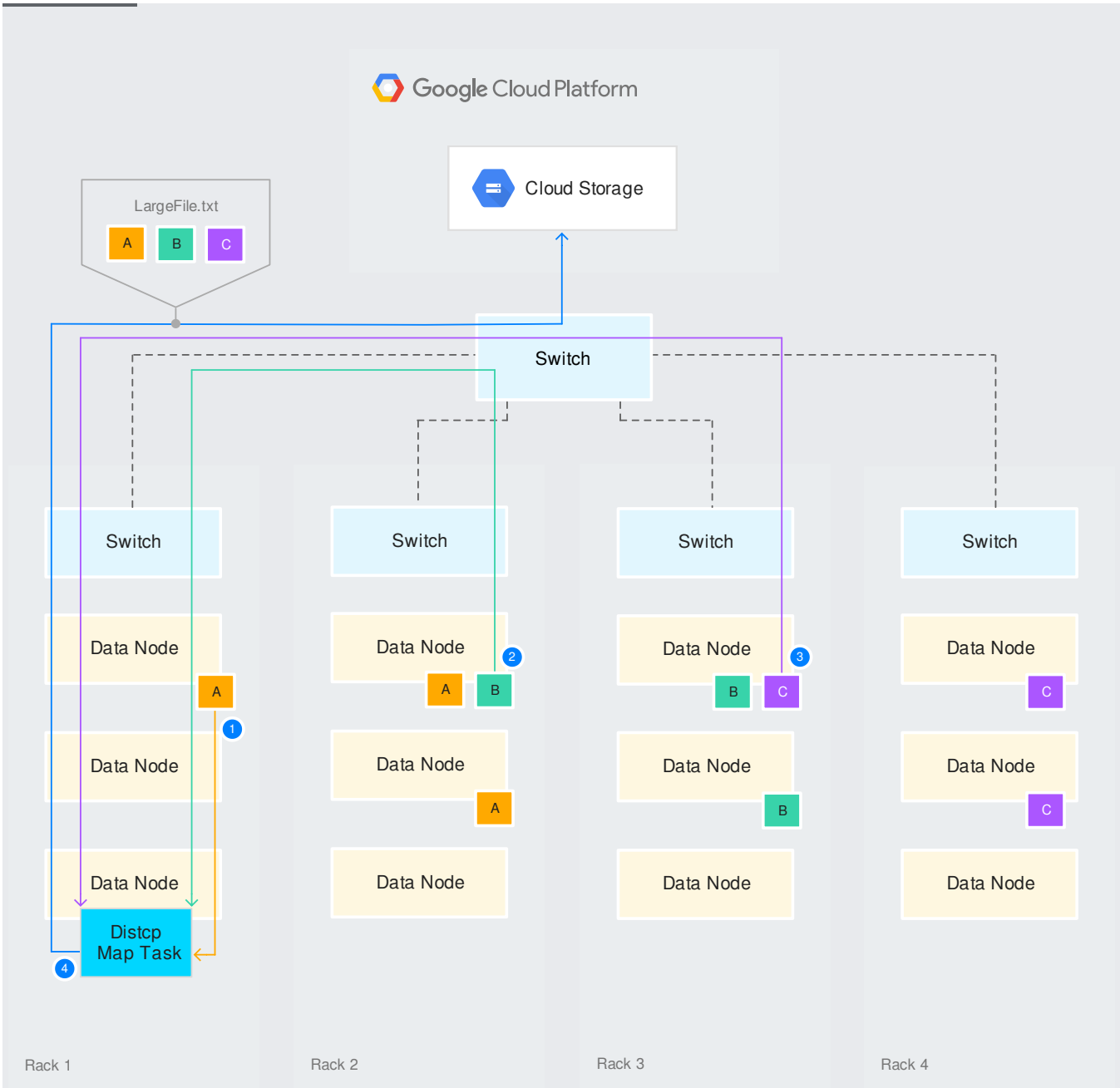
- Traffic on the source cluster's network is reduced, which allows for higher outbound bandwidths and faster transfers.
- There is no need to install the Cloud Storage connector (/dataproc/docs/concepts/connectors/cloud-storage) on the source cluster as the ephemeral Dataproc cluster, which already has the connector installed, handles the data transfer to Cloud Storage.

To understand the implications for network usage for both models, consider how Hadoop handles data replication in HDFS. Hadoop splits each file into multiple blocks — the block size is usually 128-256 megabytes. Hadoop replicates those blocks across multiple data nodes and across multiple racks to avoid losing data in the event of a data node failure or a rack failure. The standard configuration is to store 3 replicas of each block.

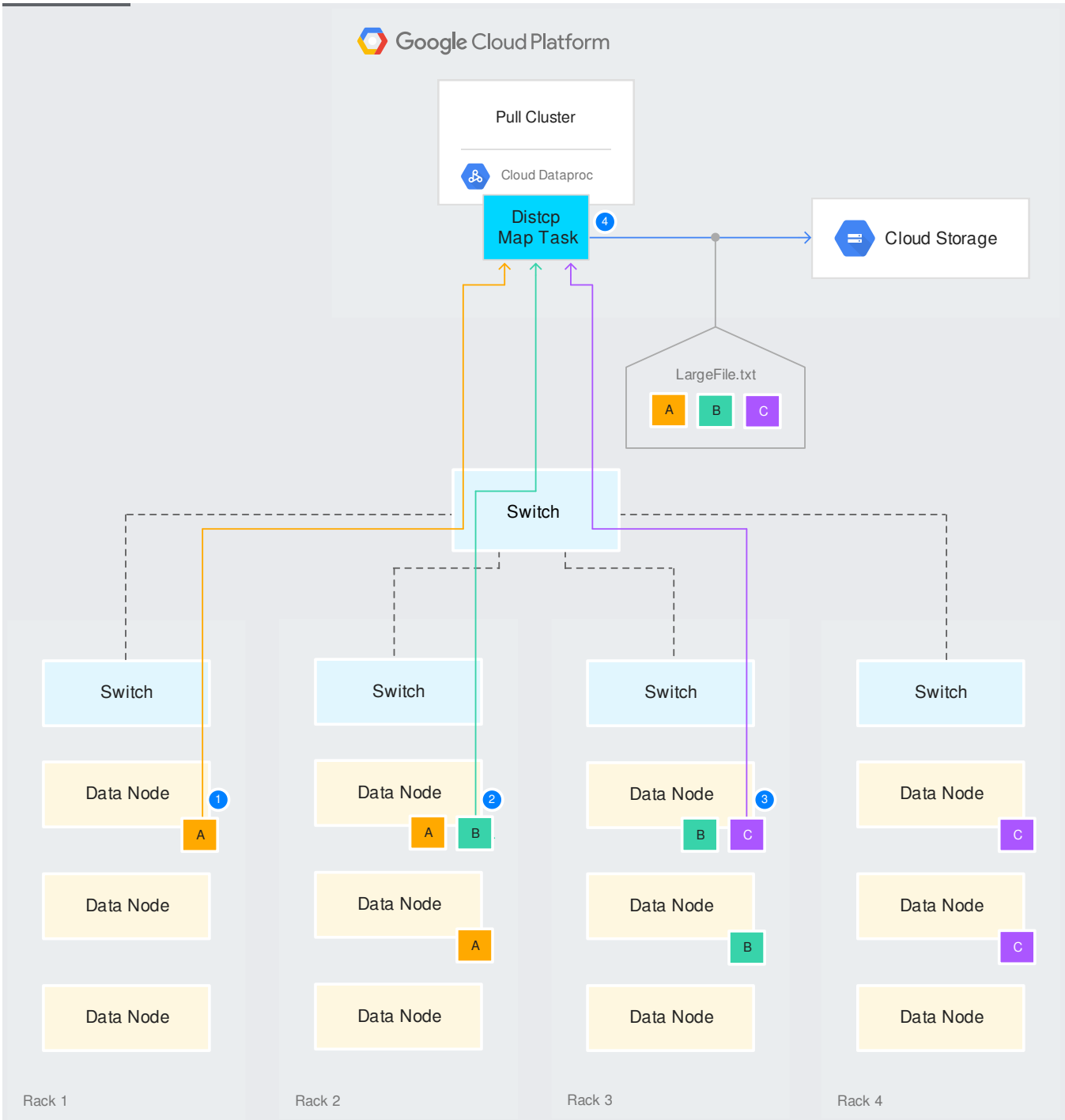
Hadoop also employs "rack awareness", which ensures that 2 copies of each block are in different data nodes inside the same rack for lower latency, and a third copy in a different rack for increased redundancy and availability. This replication logic is summarized in the following diagram:



When copying a file using the push model, that is, when the `distcp` map task runs on a data node of the source cluster, all of the file's blocks are first collected from the various data nodes. The file's blocks are then pushed out of the source cluster and over to Cloud Storage. Traffic on the network could take up to nearly twice the file's total size, as illustrated in the following diagram:



When you copy a file using the pull model (that is, when the `distcp` map task runs on a data node of the pull cluster in Google Cloud), each block travels over the network only once by exiting the source cluster directly. The overall network traffic is limited to the file's total size, as illustrated in the following diagram:



When you use the pull model, you should monitor the number of `distcp` map tasks and bandwidth used to avoid overwhelming the source cluster with too many parallel connections.

The end result of your Hadoop migration can be a cloud-native solution or a hybrid solution. The difference between these is whether your system will retain any on-premises components. In a cloud-

native solution, you house your data in the cloud and run jobs against it there. In a hybrid solution, some of your data remains on-premises. You might run jobs against that data on-premises as well, or you might run jobs in the cloud that work with on-premises data.

A cloud-native solution is the easiest to maintain, but you might have business or technical requirements that keep some data or processing on-premises. Every hybrid solution is highly case-dependent, including its own mix of technologies and services to meet the needs of your workload.

Move most of your data to Cloud Storage. However, there are some cases where you might consider moving data to a different Google Cloud product:

- If you are migrating data from Apache HBase, consider moving it to Cloud Bigtable (/bigtable/docs), which provides equivalent features.
- If you are migrating data from Apache Impala, consider moving it to BigQuery (/bigquery/docs), which provides equivalent features.

You might have data in HBase or Impala that you can use without storing it in Bigtable or BigQuery. If your job doesn't require the features of Bigtable or BigQuery, store the data in Cloud Storage.

Google Cloud doesn't use the same fine-grained permissions for files that you can achieve with HDFS on-premises. The least complicated approach to file permissions is to set them at the level of each Cloud Storage bucket. If you've set different permissions for different sets of HDFS data, consider creating different buckets in Cloud Storage that each have different permissions. Then put the HDFS data into the bucket that has the proper permissions for that data.

If you move files to a structure that's different in Cloud Storage than it is in HDFS, remember to keep track of all of the changes. When you move your jobs to Dataproc you'll need to provide the right paths to your data in its new locations.

Plan on moving your data in discrete chunks over time. Schedule plenty of time to move the corresponding jobs to the cloud between data moves. Start your migration by moving low-priority data, such as backups and archives.

If you plan to move your data incrementally, you must split your data into multiple parts. The following sections describe the most common strategies for splitting datasets.

A common approach to splitting data for an incremental move is to store older data in the cloud, while keeping your new data in HDFS in your on-premises system. This enables you to test new and migrated jobs on older, less critical data. Splitting your data in this way enables you to start your move with small amounts of data.

Important considerations:

- Can you split your data using another method in addition to splitting by time? You can get a more targeted set of data by splitting data by the jobs it supports or the organization that owns it and then splitting it further by time.
- Should you use an absolute date/time or a relative date/time? Sometimes it makes sense to split data at a point in time, such as archiving all data generated before an important change in your system. Using an absolute date/time is often appropriate if you want to create backfilling jobs to test your system in the cloud to see if you can process old data to bring it up to your current standards. In other cases, you might want to move data to the cloud based on a timestamp relative to the current date. For example, you might move all data that was created more than a year ago, or all data that hasn't been edited in the last three months.
- What date/time value are you using to make the decision? Files often have multiple date/time values. Sometimes the file creation date is the most important. Other times you might want to use the last edited time, or another timestamp from the file's metadata.
- Does all of your data have the same timestamp format? There are many ways to handle timestamps. If your data comes from more than one source, it's possible that the timestamps are stored in different formats. Ensure that you have consistent timestamps before using them to split your data.

Sometimes you can split your data by looking at the jobs that use it. This can be especially helpful if you are migrating jobs incrementally. You can move just the data that is used by the jobs that you move.

Important considerations:

- Are the jobs you are moving to the cloud self-contained? Splitting by jobs is a great option for jobs that work on self-contained units of data, but can become hard to manage if the jobs work

with data that is spread out across your system.

- Is the data likely to have the same uses in the future? Think carefully before isolating data if you might use it in different ways.
- Is the job migration scoped appropriately to result in manageable chunks of data?

You can also use broader functional criteria to split data instead of just sets of jobs. For example, you could run all of your ETL (https://wikipedia.org/wiki/Extract,_transform,_load) work in the cloud and then run analysis and reporting jobs on-premises.

In some cases, you can split your data by areas of ownership. One advantage of taking this approach is that your data organization aligns with the organization of your business. Another advantage is that it allows you to leverage Google Cloud role-based access management (/iam/docs/understanding-roles). For example, migrating data used by a particular business role to an isolated Cloud Storage bucket makes it easier to set up permissions.

Important considerations:

- Are the boundaries between owners clear? It's usually clear who the primary owner of a given data item is, sometimes the people who most often access data are not the owners.
- Are there other splitting criteria you can combine with ownership? You might end up with very large datasets after splitting by ownership. It can be useful to narrow things down even more by splitting the data again by task or by time.

One of the challenges of using a hybrid solution is that sometimes a job needs to access data from both Google Cloud and from on-premises systems. If a dataset must be accessed in both environments, you need to establish a primary storage location for it in one environment and maintain a synchronized copy in the other.

The challenges of synchronizing data are similar, regardless of the primary location you choose. You need a way to identify when data has changed and a mechanism to propagate changes to the corresponding copies. If there is a potential for conflicting changes, you also need to develop a strategy for resolving them.

Important considerations:

- Always make copies of data read-only if possible. Your synchronization strategy becomes more complex with every potential source of new edits.
- In a hybrid solution, avoid maintaining more than two copies of data. Keep only one copy on premises and only one in Google Cloud.

You can use technologies such as Apache Airflow (https://airflow.apache.org/) to help manage your data synchronization.

The following sections outline considerations for moving your data to Google Cloud.

File permissions work differently on Cloud Storage than they do for HDFS. Before you move your data to Cloud Storage, you need to become familiar with Cloud Identity and Access Management (/iam/docs) (Cloud IAM).

The easiest way to handle access control is to sort data into Cloud Storage buckets based on access requirements and configure your authorization policy at the bucket level. You can assign roles that grant access to individual users or to groups. Grant access by groups, and then assigning users to groups as needed. You need to make data access decisions as you import and structure your data in Cloud Storage.

Every Google Cloud product has its own permissions and roles. Make sure you understand the relationships between Cloud Storage access controls and those for Dataproc. Evaluate the policies for each product separately. Don't assume that the roles and permissions map directly between products.

Familiarize yourself with the following documentation to prepare for making policy decisions for your cloud-based Hadoop system:

- Overview of Cloud IAM for Cloud Storage (/storage/docs/access-control/iam)
- List of Dataproc permissions and IAM roles (/dataproc/docs/concepts/iam/iam)

If you need to assign more granular permissions to individual files, Cloud Storage supports access control lists (/storage/docs/access-control/lists) (ACLs). However, Cloud IAM is the strongly preferred option. Only use ACLs if your permissions are particularly complex.

Because Cloud Storage is a Hadoop compatible file system, you can use Hadoop DistCp (https://hadoop.apache.org/docs/current/hadoop-distcp/DistCp.html) to move your data from your on-premises HDFS to Cloud Storage. You can move data several ways using DistCp. We recommend this way:

1. Use Cloud VPN (/vpn/docs/concepts/overview) to establish a virtual private network tunnel between your Google Cloud project and your on-premises network.
2. Create a Dataproc cluster (/dataproc/docs/guides/create-cluster) to use for the data transfer.
3. Use the gcloud command-line tool to connect to your cluster's master instance. For example:

   Where `CLUSTER_NAME` is the name of the Dataproc cluster you created for the job. The suffix `-m` identifies the master instance.
4. On the cluster's master instance, run DistCp commands to move the data.

The actual DistCp commands you need to move your data are similar to the following:

In this example `nn1` and `8020` are the namenode and port where your source data is stored, and `bucket` is the name of the Cloud Storage bucket that you are copying the file to.

When you're copying or moving data between distinct storage systems such as multiple HDFS clusters or between HDFS and Cloud Storage, it's a good idea to perform some type of validation to guarantee data integrity. This validation is essential to be sure data wasn't altered during transfer. For more details, refer to the guide on Validating data transfers (/solutions/migration/hadoop/validating-data-transfers).

Cloud Storage maintains an index of object keys for each bucket in order to provide consistent object listing. This index is stored in lexicographical order and is updated whenever objects are written to or deleted from a bucket. Adding and deleting objects whose keys all exist in a small range of the index naturally increases the chances of contention.

Cloud Storage detects such contention and automatically redistributes the load on the affected index range across multiple servers. If you're writing objects under a new prefix and anticipate that you will get to a rate greater than 1000 write requests per second, you should ramp up the request rate gradually, as described in the Cloud Storage documentation (https://cloud.google.com/storage/docs/request-rate#ramp-up). Not doing so may result in temporarily higher latency and error rates.

The simplest way to transfer data from your on-premises clusters to Google Cloud is to use a VPN tunnel (/vpn/docs/concepts/overview) over the public internet. If a single VPN tunnel doesn't provide the necessary throughput, multiple VPN tunnels (/solutions/building-high-throughput-vpns) might be created and Google Cloud will automatically distribute traffic across tunnels to provide additional bandwidth.

Sometimes even multiple VPN tunnels don't provide sufficient bandwidth or connection stability to meet the needs of your migration. Consider the following approaches to improve throughput:

- Use direct peering between your network and Google's edge points of presence (PoPs) (/cdn/docs/locations). This option reduces the number of hops between your network and Google Cloud.
- Use a Cloud Interconnect service provider (/interconnect/docs/concepts/service-providers) to establish a direct connection to Google's network. The service details vary for different partners. Most offer an SLA for network availability and performance. Contact a service provider directly to learn more.

Google Cloud works with a variety of partners that can assist you in migrating your data. Check out the partners working with Cloud Storage (https://cloud.google.com/storage/partners/) for more information about services available to help you with your data migration. The available services and terms vary by provider, so work with them directly to get details.

- 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/).
- Try out other Google Cloud features for yourself. Have a look at our tutorials (/docs/tutorials).