# Tiering strategies

## Enterprise applications

NetApp
February 10, 2026

# Table of Contents


Tiering strategies . . . 1
- Full file tiering . . . 1
  - Policies . . . 1
- Partial file tiering . . . 1
  - Policies . . . 2
- Archive log tiering . . . 2
  - Policies . . . 2
- Snapshot tiering . . . 3
  - Policies - local snapshots . . . 3
  - Policies - replicated snapshots . . . 3
- Backup tiering . . . 3

# Tiering strategies

## Full file tiering

Although FabricPool tiering operates at the block level, in some cases it can be used to provide file-level tiering.

Many applications datasets are organized by date, and such data is generally increasingly less likely to be accessed as it ages. For example, a bank might have a repository of PDF files that contain five years of customer statements, but only the most recent few months are active. FabricPool can be used to relocate older datafiles to the capacity tier. A cooling period of 14 days would ensure the more recent 14 days of PDF files remain on the performance tier. Furthermore, files that are read at least every 14 days would remain hot and therefore remain on the performance tier.

### Policies

To implement a file-based tiering approach, you must have files that are written and not subsequently modified. The `tiering-minimum-cooling-days` policy should be set high enough so that files that you might need remain on the performance tier. For example, a dataset for which the most recent 60 days of data is required with optimal performance warrants setting the `tiering-minimum-cooling-days` period to 60. Similar results can also be achieved based on the file access patterns. For example, if the most recent 90 days of data is required and the application is accessing that 90-day span of data, then the data would remain on the performance tier. By setting the `tiering-minimum-cooling-days` period to 2, you get prompt tiering after the data becomes less active.

The `auto` policy is required to drive tiering of these blocks because only the `auto` policy affects blocks that are in the active file system.

Any type of access to data resets the heat map data. Virus scanning, indexing, and even backup activity that reads the source files prevents tiering because the required `tiering-minimum-cooling-days` threshold is never reached.

## Partial file tiering

Because FabricPool works at the block level, files that are subject to change can be partially tiered to object storage while also remaining partially on performance tier.

This is common with databases. Databases that are known to contain inactive blocks are also candidates for FabricPool tiering. For example, a supply chain management database might contain historical information that must be available if needed but is not accessed during normal operations. FabricPool can be used to selectively relocate the inactive blocks.

For example, datafiles running on a FabricPool volume with a `tiering-minimum-cooling-days` period of 90 days retains any blocks accessed in the preceding 90 days on the performance tier. However, anything that is not accessed for 90 days is relocated to the capacity tier. In other cases, normal application activity preserves the correct blocks on the correct tier. For example, if a database is normally used to process the previous 60 days of data on a regular basis, a much lower `tiering-minimum-cooling-days` period can be set because the natural activity of the application makes sure that blocks are not relocated prematurely.

The `auto` policy should be used with care with databases. Many databases have periodic activities such as end-of-quarter process or reindexing operations. If the period of these operations is greater than the `tiering-minimum-cooling-days` performance problems can occur. For example, if end-of-quarter processing requires 1TB of data that was otherwise untouched, that data might now be present on the capacity tier. Reads from the capacity tier is often extremely fast and may not cause performance problems, but the exact results will depend on the object store configuration.

### Policies

The `tiering-minimum-cooling-days` policy should be set high enough to retain files that might be required on the performance tier. For example, a database in which the most recent 60 days of data might be required with optimal performance would warrant setting the `tiering-minimum-cooling-days` period to 60 days. Similar results could also be achieved based on the access patterns of files. For example, if the most recent 90 days of data is required and the application is accessing that 90- day span of data, then the data would remain on the performance tier. Setting the `tiering-minimum-cooling-days` period to 2 days would tier the data promptly after the data becomes less active.

The `auto` policy is required to drive tiering of these blocks because only the `auto` policy affects blocks that are in the active file system.

Any type of access to data resets the heat map data. Therefore, database full table scans and even backup activity that reads the source files prevents tiering because the required `tiering-minimum-cooling-days` threshold is never reached.

## Archive log tiering

Perhaps the most important use for FabricPool is improving the efficiency of known cold data, such as database transaction logs.

Most relational databases operate in transaction log archival mode to deliver point-in-time recovery. Changes to the databases are committed by recording the changes in the transaction logs, and the transaction log is retained without being overwritten. The result can be a requirement to retain an enormous volume of archived transaction logs. Similar examples exist with many other application workflows that generate data that must be retained, but is highly unlikely to ever be accessed.

FabricPool solves these problems by delivering a single solution with integrated tiering. Files are stored and remain accessible in their usual location, but take up virtually no space on the primary array.

### Policies

Use a `tiering-minimum-cooling-days` policy of a few days results in retention of blocks in the recently created files (which are the files most likely to be required in the near term) on the performance tier. The data blocks from older files are then moved to the capacity tier.

The `auto` enforces prompt tiering when the cooling threshold has been reached irrespective of whether the logs have been deleted or continue to exist in the primary file system. Storing all the potentially required logs in a single location in the active file system also simplifies management. There is no reason to search through snapshots to locate a file that needs to be restored.

Some applications, such as Microsoft SQL Server, truncate transaction log files during backup operations so

that the logs are no longer in the active file system. Capacity might be saved by using the `snapshot-only` tiering policy, but the `auto` policy is not useful for log data because there should rarely cooled log data in the active file system.

# Snapshot tiering

The initial release of FabricPool targeted the backup use case. The only type of blocks that could be tiered were blocks that were no longer associated with data in the active file system. Therefore, only the snapshot data blocks could be moved to the capacity tier. This remains one of the safest tiering options when you need to ensure performance is never affected.

## Policies - local snapshots

Two options exist for tiering inactive snapshot blocks to the capacity tier. First, the `snapshot-only` policy only targets the snapshot blocks. Although the `auto` policy includes the `snapshot-only` blocks, it also tiers blocks from the active file system. This might not be desirable.

The `tiering-minimum-cooling-days` value should be set to a time period that makes data that might be required during a restoration available on the performance tier. For example, most restore scenarios of a critical production database include a restore point at some time in the previous few days. Setting a `tiering-minimum-cooling-days` value of 3 would make sure that any restoration of the file results in a file that immediately delivers maximum performance. All blocks in the active files are still present on fast storage without needing to recover them from the capacity tier.

## Policies - replicated snapshots

A snapshot that is replicated with SnapMirror or SnapVault that is only used for recovery should generally use the FabricPool `all` policy. With this policy, metadata is replicated, but all data blocks are immediately sent to the capacity tier, which yields maximum performance. Most recovery processes involve sequential I/O, which is inherently efficient. The recovery time from the object store destination should be evaluated, but, in a well-designed architecture, this recovery process does not need to be significantly slower than recovery from local data.

If the replicated data is also intended to be used for cloning, the `auto` policy is more appropriate, with a `tiering-minimum-cooling-days` value that encompasses data that is expected to be regularly used in a cloning environment. For example, a database's active working set might include data read or written in the previous three days, but it could also include another 6 months of historical data. If so, then the `auto` policy at the SnapMirror destination makes the working set available on the performance tier.

# Backup tiering

Traditional application backups include products such as Oracle Recovery Manager, which create file-based backups outside the location of the original database.

`tiering-minimum-cooling-days` policy of a few days preserves the most recent backups, and therefore the backups most likely to be required for an urgent recovery situation, on the performance tier. The data blocks of the older files are then moved to the capacity tier.

The `auto` policy is the most appropriate policy for backup data. This ensures prompt tiering when the cooling threshold has been reached irrespective of whether the files have been deleted or continue to exist in the primary file system. Storing all the potentially required files in a single location in the active file system also simplifies management. There is no reason to search through snapshots to locate a file that needs to be restored.

The `snapshot-only` policy could be made to work, but that policy only applies to blocks that are no longer in the active file system. Therefore, files on an NFS or SMB share must be deleted first before the data can be tiered.

This policy would be even less efficient with a LUN configuration because deletion of a file from a LUN only removes the file references from the file system metadata. The actual blocks on the LUNs remain in place until they are overwritten. This situation can create a lengthy delay between the time a file is deleted and the time that the blocks are overwritten and become candidates for tiering. There is some benefit to moving the `snapshot-only` blocks to the capacity tier, but, overall, FabricPool management of backup data works best with the `auto` policy.

This approach helps users manage the space required for backups more efficiently, but FabricPool itself is not a backup technology. Tiering backup files to object store simplifies management because the files are still visible on the original storage system, but the data blocks in the object store destination depend on the original storage system. If the source volume is lost, the object store data is no longer useable.

## Copyright information



## Trademark information