NETAPP TECHNICAL REPORT

Using FlexClone to Clone Files and LUNs

Uday Boppana, NetApp
March 2010 | TR-3742

FLEXCLONE ENHANCEMENTS TO ENABLE FILE AND LUN CLONING

FlexClone® functionality in Data ONTAP® 7.3.1 has been enhanced with the ability to clone individual files in NAS and SAN environments. There is also a newer way to clone LUNs without the need for a backing Snapshot™ copy. This document gives a detailed description of FlexClone at the file and LUN level.

TABLE OF CONTENTS

1 INTRODUCTION

NetApp® FlexClone technology is enhanced in Data ONTAP 7.3.1 to provide space-efficient cloning at different granularities. FlexClone technology now gives the user the ability to clone individual files that are present in a FlexVol® volume in a NAS environment or inside a LUN in a SAN environment. There is also a newer way to clone LUNs without the need for a backing Snapshot copy. This document gives a detailed description of FlexClone at the file and LUN level.

FlexClone at the FlexVol level has been in Data ONTAP since Data ONTAP 7.0. A FlexClone volume is a writable point-in-time image based on Snapshot of a FlexVol volume or another FlexClone volume. FlexClone volumes use space very efficiently, leveraging the Data ONTAP architecture to store only data that changes between the source and clone. For more information on FlexClone at the FlexVol level, refer to NetApp technical report TR-3374.

Starting with Data ONTAP 7.3.1, you can create a clone of file that is present either in a FlexVol volume in a NAS environment or inside a LUN in a SAN environment. Using the enhanced FlexClone technology, LUNs can be cloned without the need for a backing Snapshot copy.

Cloning of files and LUNs using FlexClone technology is highly space efficient since the cloned copies share the same physical data space with the source and occupy negligible extra space in the storage system for their initial metadata. Cloned files or LUNs start occupying extra space only when the data in the source or the clone is overwritten. FlexClone creation is also a fast and time-efficient process since there is no physical copy of data involved. Cloning using FlexClone at file and LUN level is Snapshot independent and so does not need a backing Snapshot copy.

The process of creating a clone of an existing LUN or file has no impact on the client access to the source file or LUN either during the creation process or after the cloning is done. Clients that are accessing the source file or LUN will experience no disruption or outage during the clone creation process and can write to the source file or LUN while the cloning process is in progress. Once the cloning process is complete, the clone files or LUNs can be accessed from the client and all operations performed just like regular files and LUNs.

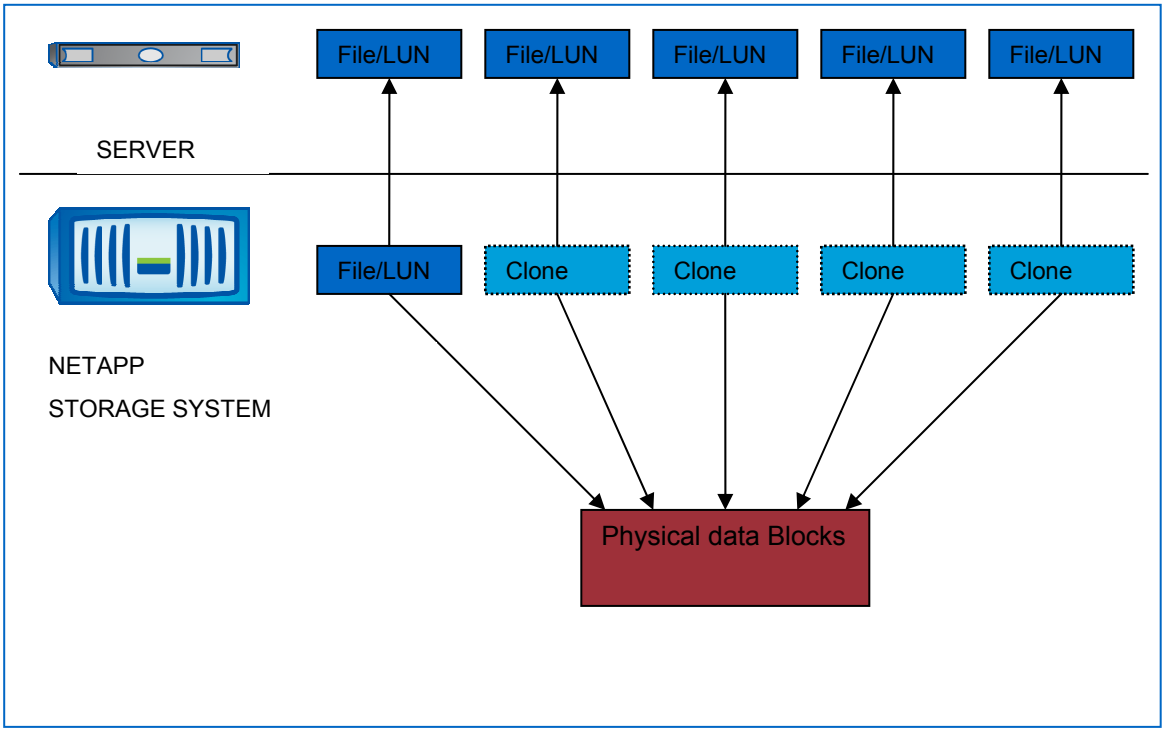


Figure 1) The source file or LUN and the clone files or LUNs appear to the client systems as regular files or LUNs but share the physical storage space on the storage system.

The clients see the clone files or LUNs as regular files and LUNs. All regular file and LUN operations are supported on both the source and the clones. If a file or LUN has clone files or LUNs, either the source file or LUN or any of the clone files or LUNs can be deleted with no effect on the source file or the clone files.

The new cloning abilities combined with the existing functionality to create FlexClone copies of FlexVol volumes provide a space- and time-efficient solution for many data center problems that involve multiple copies of the same data set and drastically reduce the storage space needed to store duplicate copies of data. FlexClone at the FlexVol volume, file, and LUN levels can be combined to create a powerful time- and space-efficient solution to store redundant data sets since all the redundant files or LUNs share the same underlying physical storage.

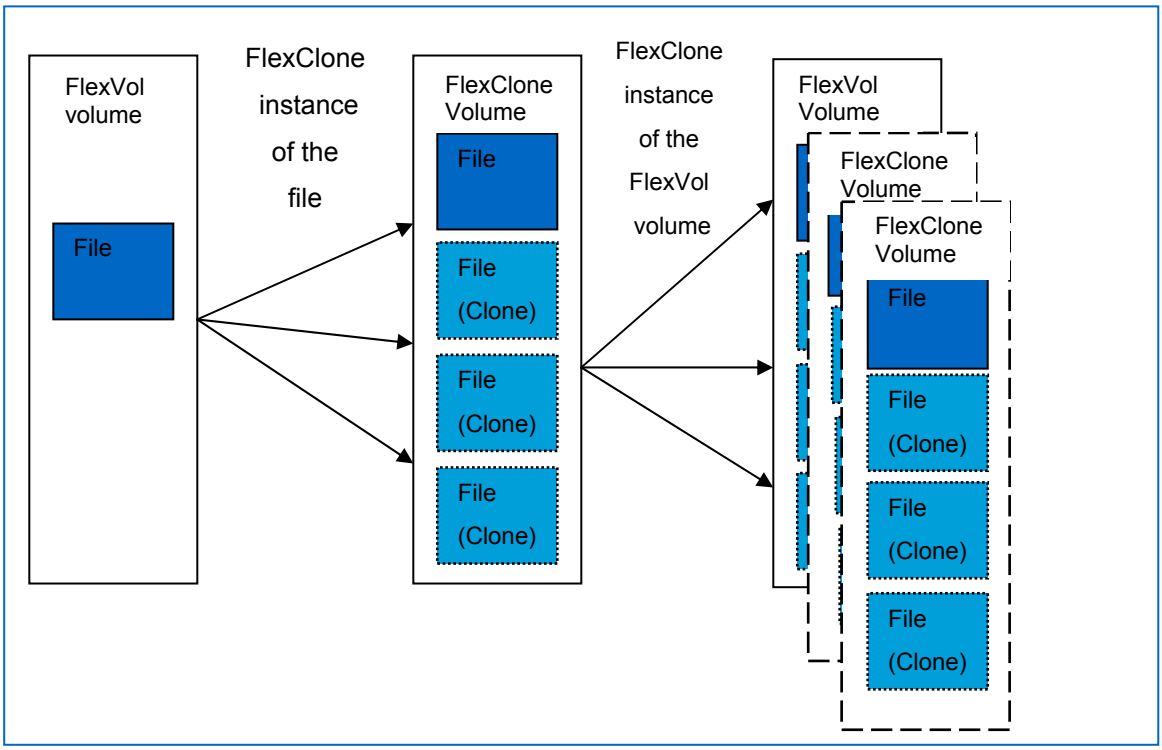


Figure 2) Combining FlexClone at FlexVol and file or LUN level to create multiple copies of the same file or LUN all based on the same physical storage.

As shown above, we can use FlexClone at the file or LUN level to create clone files or LUNs within the same FlexVol volume, all of which share the same underlying storage. Then we can use FlexClone at the FlexVol level to create FlexClone volumes of the parent FlexVol volume. The FlexClone volumes share the data blocks with the parent FlexVol volume. So, we have lots of copies of the same file or LUN all sharing the same underlying physical storage. These clone files or LUNs start occupying extra space only when data in them is overwritten.

2 CONFIGURATION AND OPERATION

This section gives an overview of the requirements for enabling FlexClone at the file and LUN level and the process of creating a file or LUN clone on a NetApp storage system.

2.1 PLATFORMS

FlexClone at the file and LUN granularities is supported on the following platforms:

- FAS2000 series
- FAS3000 series
- FAS3100 series

- FAS6000 series
- NearStore® R200

2.2 DATA ONTAP

The enhanced ability of FlexClone to clone files and LUNs without the need for a backing Snapshot copy is available in Data ONTAP 7.3.1 and later.

2.3 LICENSING

The enhanced ability of FlexClone to clone files and LUNs without the need for a backing Snapshot copy, along with the existing ability to create FlexClone volume of a FlexVol volume, is available under the FlexClone feature license. You need to license `flex_clone` to enable the FlexClone feature on the storage system using the following command:

```
License add <<flex_clone license key>>
```

In an HA pair environment, you need to add the license on both the nodes to enable the feature on both of them.

If you already have a FlexClone license and are running an older version of Data ONTAP, then upgrading to a Data ONTAP version that supports FlexClone at file and LUN level will give you access to this enhanced functionality.

2.4 CREATING A FLEXCLONE INSTANCE OF A FILE IN A NAS ENVIRONMENT

FlexClone at the file level can be used to create clones of individual files stored in a FlexVol volume in a NetApp storage system for both NFS and CIFS environments. The clone file must reside in the same FlexVol volume as the source file. All the cloned files refer to the same physical data blocks on the disk. There is a minimal amount of space overhead for each clone for storing its metadata. A FlexClone instance of a file can be created in a NAS environment with the following steps:

- Issue a `clone start` command with source and destination file paths. The source and destination paths must be in the same FlexVol volume. Once the clone operation is successfully started, Data ONTAP will print the ID of the clone operation that was started. The source file can be accessed by the client and can be written to while the cloning operation is in progress.
- In order to create a point-in-time image of the source file or LUN, clone command by default creates a temporary Snapshot copy of the FlexVol volume in which the clone is being created. This is done so that the clone is a point-in-time image of the source file or LUN and that any active writes to the source file do not affect the process of clone creation. If the file being cloned is not being actively written to, then the creation of temporary Snapshot copy can be stopped by using the `-n` flag of the clone command. It is not recommended to prevent the temporary Snapshot copy creation unless you are absolutely sure that there is no write or other data modification activity to the source.
- The `clone status` command can be used to query the storage system for the status of the clone operation. The time to create the clone is dependent on the size of the source file, and in most cases it's a fast operation since there is no physical copying of any blocks involved.
- Once the cloning process is completed, a message is printed on the console informing the user of the completion. A message informing the completion is also logged in the messages file and an EMS event is generated and logged in the EMS log of the storage system. The temporary Snapshot copy that was created before the start of the clone operation is deleted.
- The clone file is now ready to be used by NFS or CIFS clients. The clients see the clone and source as separate files. These source and clone files can also be accessed by any of the data access protocols that can access files, including http, ftp, and NDMP.
- Note that there is no interruption to any read, write, or other operations on the source file while a clone operation is in progress.

2.5 CREATING A FLEXCLONE INSTANCE OF A LUN

FlexClone can be used to clone a complete LUN that is present in a FlexVol volume on the storage system. The process to clone a LUN is similar to cloning a file. The clone LUN must reside in the same FlexVol volume as the source LUN. Data ONTAP has the ability to recognize the source as a LUN and takes the necessary steps to clone a LUN. Though the process to share physical blocks between a LUN and its clone is similar to file cloning, cloning a LUN involves extra metadata operations that must be performed to present the LUN to the SAN subsystem of Data ONTAP. Data ONTAP automatically performs these operations in the background when the source is a LUN. Also, a FlexClone instance of a LUN requires slightly more space for its metadata due to the SAN subsystem-related metadata. Once the cloning is complete, the clone LUN is ready to be presented to a SAN client.

Note that Data ONTAP has had the ability to clone a LUN using a backing Snapshot copy since Data ONTAP 7.0. This method of creating a LUN clone still exists and is supported. Using FlexClone to clone a LUN is an enhancement that is introduced in Data ONTAP 7.3.1 and is Snapshot copy independent.

2.6 CREATING A FLEXCLONE INSTANCE OF A FILE IN A SAN ENVIRONMENT

FlexClone can be used to clone files that are present inside a LUN in a SAN environment. Data ONTAP provides the API to be able to do this. However, host-side support is needed to integrate the clone file into the host file system and to make the clone file usable by the client. The API can be used to create a clone of a file on the storage system as follows:

- To clone a file that is present inside a LUN, the user needs to specify the address of the data blocks of the file in logical block addressing (LBA) format as seen by the host. The –r flag of the `clone start` command must be used to specify the source and destination LBA addresses and the count of number of data blocks to be cloned.
- The rest of the process of clone creation is the same as the process outlined in the NAS section above except that the `clone start` command must be used with the `-r` flag to start the process.
- Host-side tool support is needed to integrate the cloned data blocks into the host file system and to present the cloned file to the host file system.

2.7 DELETING THE SOURCE FILE OR LUN

Once clone files or LUNs have been created, the source file or LUN or any of the clones can be deleted. Deleting the source file or LUN has no impact on the clone files or LUNs. All the clone files or LUNs still exist and show up to the clients as normal files or LUNs. Similarly, deleting a clone file or LUN has no impact on the other clones or the source. The clone LUNs or files still exist and they still share the same physical data blocks on the disk.

When a file or LUN or its clone that has shared blocks is deleted, the shared blocks continue to be used by the remaining clone files or LUNs. So, deleting a clone will only free up the space that is being used by its metadata. No space used by the shared data blocks is returned to free space until the source file or LUN and all clone files or LUNs of that source are deleted. Once a source file or LUN and all its clone files or LUNs are deleted, the data blocks are freed and the space returned to the free space pool.

2.8 OVERWRITING THE DATA IN THE SOURCE OR CLONE FILE OR LUN

When a clone file or LUN is initially created, both the source and the clone share the same data blocks. Once data in either the source or the clone is overwritten with new data, more space will start to be consumed since the clone and the source do not share the newly written data. The newly written data is stored separately for each individual entity, the source and the clone. Note that even if the same data is written both the source and the clone, they still will point to different blocks on disk and will not share the physical blocks. The physical blocks are only shared immediately after the cloning process. If the newly written data to the source and clone is similar, deduplication can be run on the FlexVol volume that contains the clone to get further space savings since deduplication can eliminate duplicate blocks.

2.9 VOLUME STATUS

The `vol status` command of Data ONTAP shows the status of all volumes that contain file and LUN clones created using the `clone` command as having the attribute "`sis`". The `sis` attribute is added to the

FlexVol volume when the first FlexClone instance of a file or LUN is created. Note that volumes that have deduplication enabled on them also show up with the attribute `sis` in the output of `vol status` command.

2.10 VIEWING THE SPACE SAVINGS DUE TO FLEXCLONE FILE OR LUN

The "`df –s`" command with the FlexVol volume name as argument will list how much space in the volume has been saved because the clone files or LUNs share the same physical blocks instead of each having their own physical copy of the blocks. Note that if deduplication is enabled on the same FlexVol volume that has FlexClone files or LUNs, the output of "`df –s`" will display the total saving due to the combination of both deduplication and FlexClone. The "`df –s`" command displays the space savings due to FlexClone as a percentage of the sum of the used and saved space on the FlexVol volume.

For example, if you have a FlexVol volume of size 100GB with 50GB used space and then create a file of 10GB and then create one clone of it, the total used physical space is approximately 60GB (50GB + 10GB for file and its clone). If the clone was a full physical copy, you would have used 70GB (50GB + 10GB for file + 10GB for the clone). So, we saved space of 10GB by cloning. So the savings are 14% ((10/70)*100).

2.11 MONITORING THE CLONE OPERATION

The "`clone status`" command can be used to monitor the status of a clone operation. For clone operations that are currently in progress, the command will report, among other things, the percentage of the clone operation completed. For clone operations that were terminated unsuccessfully, it will report the reason that caused the clone operation to terminate. The "`clone status`" command will not display the status of any successfully completed clone operations.

Clone operation can terminate unsuccessfully for a variety of reasons. The conditions that can lead to unsuccessful termination are listed in this document in different sections. Creating a FlexClone instance of a file in NAS environments and creating a FlexClone instance of a LUN is an atomic operation, and so if the creation of the clone terminates unsuccessfully in the middle of the creation process, all the changes that were made till that point are undone, and the partially created clone is removed. However, in the case of cloning a file that is present inside a LUN, if a cloning operation terminates unsuccessfully, the shared blocks are left as shared, and the changes are not rolled back. The host application that initiated the clone operation must take necessary steps to make sure it deletes the partially cloned file. In either case, a message is printed on the storage system console indicating the unsuccessful termination of the clone operation. A message is also logged in the messages file, and an EMS event is generated and also logged in the EMS log. A small amount of metadata is stored in a metadata file on the storage system to indicate the unsuccessful termination of the clone creation and the reason for the termination. Any temporary Snapshot copy that was created prior to starting the clone operation is deleted. A `clone status` command will indicate the clone's operation as failed and the reason for the failure. A `clone clear` command will clear the failure status metadata stored on disk. NetApp best practice recommendation is to clear the status of unsuccessfully terminated clone operations once the cause for termination is understood.

2.12 COMMAND SUMMARY

Here is a brief summary of the clone command and its subcommands. For a more detailed description of the commands, refer to the Data ONTAP documentation on NOW™ (NetApp on the Web) site at http://now.netapp.com/.

Command	Purpose
`clone start <src path> <dest_path> [-n] [-l]`	Starts the process of creating a FlexClone instance of the file or LUN. The file or LUN at source indicated by the src_path argument is cloned to the destination path indicated by the dest_path argument. The clone file or LUN must be in the same FlexVol volume as the source. Upon a successful start, this command outputs the ID of the clone operation that was started. The –n flag prevents the creation of the temporary Snapshot copy prior to starting the clone process.

	The –l flag will enable change logging for clone data blocks. This flag can only be used when creating a clone inside a FlexVol volume that already has deduplication enabled on it. Using this flag while creating a clone on a FlexVol volume that does not have deduplication enabled on it will result in an error. For details on change logging refer to the later sections of this document.
`clone start <src path> [dest_path] [-n] [-l] <-r <src_fbn>:<dest_fbn>:<fbn_cnt> ...>`	Starts the clone process for cloning a file present inside a LUN. For cloning a file inside a LUN, the src_fbn and dest_fbn are the LBA addresses of the data blocks in the source file and the destination address where the blocks will be referenced. The fbn_cnt parameter is the count of number of blocks starting with the source LBA address that need to be cloned. The LBA addresses that are used here are the LBA addresses within LUN as seen by the host.
`clone status [vol-name [ID]]`	Gives the status of all the active and unsuccessfully terminated clone operations on the volume. If the ID parameter is specified, it gives the clone status for that particular clone operation. Note that status of failed operations is displayed only for those operations that have the failure metadata stored on the disk. If the metadata was cleared using the `clone clear` command, no status is displayed for that operation. Also no status is displayed for successfully completed clone operations.
`clone stop <vol-name> <ID>`	Aborts the particular clone operation and deletes the temporary Snapshot copy if one was created.
`clone clear <vol-name> <ID>`	Clears the metadata stored on the disk to indicate the failure of a particular clone operation.
`man clone`	Gives the man page for the clone command.
`Manage ONTAP SDK`	The Manage ONTAP® SDK has the ONTAPI™ programming interfaces for the clone functionality. Refer to the SDK for more details.

3 BEHAVIOR OF FILE AND LUN CLONE

This section discusses in detail how file or LUN clone behaves with different options of Data ONTAP.

3.1 SPACE RESERVATION

Space reservation for FlexClone has different behavior for file clone and LUN clone.

FlexClone files are always non space reserved after creation. So, irrespective of what the space reservation on the source file is, the clone file has no space reservation on it. To enable space reservation on a file clone, use the `file reservation` command.

FlexClone LUNs inherit the space guarantee settings of their source. So, if there is not enough space in the FlexVol volume to create a FlexClone instance of a LUN with space guarantee same as that of the source, the cloning process will fail. Note that the source and clone LUN will share blocks on the disk even with space guarantee enabled.

3.2 QUOTAS

Quota usage for clones is charged at the logical level. So, the amount of extra space usage charged to the quota for creating a clone is equal to the total logical size of the clone. For example, if you create a clone of a 10GB file, the total used space charged to the quota for the source file and the clone file is 20GB (10GB for the source and 10GB for the clone).

The effect of exceeding the quota limit due the creation of a FlexClone instance of a file is different for qtree quotas and user or group quotas. If the total logical used space occupied after a clone creation will be more than the allowed tree quota for the qtree, the clone operation will fail. If the total logical used space occupied after a clone creation will be more than the allowed quota for that user or group, the clone operation will still succeed if the FlexVol volume has enough space to hold the metadata or the data for the clone. However, after the success of the clone operation, the quota for that user or group will be oversubscribed.

3.3 ACCESS CONTROL LISTS (ACL) AND STREAMS

If the file that is the source of a FlexClone operation has access control lists or streams, the ACLs and streams will not be cloned and so will not be present for the clone file. So, if you want the same ACL based permission as the source file on the clone file, or attach streams to the clone file, the storage administrator must do so separately on the clone file after the cloning process is completed.

3.4 ROLE-BASED ACCESS CONTROL (RBAC)

The access to the clone command can be controlled by using role-based access control capabilities provided by Data ONTAP using the `useradmin` command. You can create roles that have access to only the commands that are needed to use the clone functionality. Here is an example set of commands to use to enable RBAC for clone command:

Create a role that has access to all login methods but can use only the `clone` commands after logging in.

```
netapp01>useradmin role add cloneadmin -a login-*,cli-clone*,api-clone-*
```

Add a group that has the role of `cloneadmin` just created:

```
netapp01>useradmin group add cloneadmin_group -r cloneadmin
```

Add users to the group:

```
netapp01>useradmin user add cloneu1 -g cloneadmin_group
```

The user, cloneu1 in this case, will have access only to the `clone` command of Data ONTAP. For more details on roles, groups, and users, refer to the Data ONTAP system administration guide on the NOW site at http://now.netapp.com.

4 RECOMMENDATIONS

This section outlines some of the recommended steps for creating a large number of clone LUNs or files while using the least amount of space. For detailed discussion around the feature related thresholds and recommendations for the FlexClone feature, refer to the “About FlexClone files and FlexClone LUNs” section of the Data ONTAP storage management guide on NOW site at http://now.netapp.com/.

CREATING A LARGE NUMBER OF CLONE FILES OR LUNS

To create a large number of clone files or LUNs on the storage system it is recommended to combine the different granularities of FlexClone copies rather than using the same granularity.

Suppose that you want to create 2,000 clone files from a single file. Here is how it can be achieved:

1. Create 99 FlexClone files from the source file for a total of 100 logical files within the same volume.

2. Create 19 FlexClone volumes of the FlexVol volume that the files reside in for a total of 20 FlexVol volumes. We now have 2,000 logical files all sharing the same physical storage.

CREATING A LARGE NUMBER OF CLONE FILE OR LUNS IN THE SAME VOLUME

The maximum number of times a block can be cloned is 255. All clones of the block beyond that will each be a physical copy. To minimize the space consumed by the clone files, you can follow this process:

1. Create 255 clone files or LUNs.

2. Then create the 256th clone file or LUN. This 256th clone will be a full physical copy of the source file or LUN.

3. Now, create the clone files or LUNs from the 257th clone onward using the 256th clone file or LUN created in the previous step as the source until you create 255 more clone files or LUNs. In this way you can create 255 more clone files or LUNs based on the one physical copy created in the previous step. So, you have 512 logical files or LUNs all sharing two physical on disk copies. Repeat this process till you reach your target file or LUN clone count.

One important issue to consider when planning for environments with a large number of FlexClone files or LUNs is the performance requirements of the host-side application and the performance expected from the storage system. The performance section of this document has a discussion about the performance characteristics of FlexClone at file and LUN level.

5 INTEROPERABILITY WITH OTHER NETAPP PRODUCTS

5.1 VOLUME SNAPMIRROR

Volume SnapMirror® is an efficient data replication product that transfers only the modified data disk blocks to the destination after the initial base transfer. Volume SnapMirror transfers data in 4KB disk blocks to the destination. If a FlexVol volume that is the source of a volume SnapMirror copy contains FlexClone files or LUNs, then volume SnapMirror transfers only the physical block and a small amount of metadata as shown below. On the destination also only one copy of the physical block along with its metadata is stored and the block is shared among the source and its clones. So, the destination volume will look exactly like the source volume and all the clone files or LUNs on the destination will share the same physical blocks as shown below:

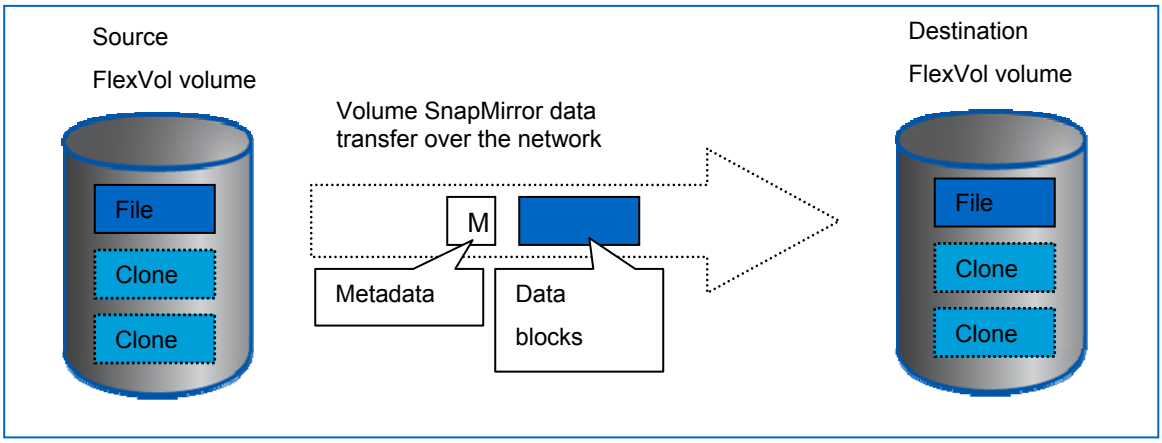


Figure 3) Volume SnapMirror transfer of FlexClone instance of file or LUN.

Volume SnapMirror is efficient and conserves network bandwidth as well as the storage space on the destination when the source has FlexClone instances of files or LUNs.

If you have a SnapMirror relationship where the size of the source FlexVol volume is smaller than the allowed volume size limit and the destination is larger than the FlexClone limit for that platform, creating a FlexClone instance of a file or LUN on the source FlexVol volume will cause the next SnapMirror update to fail since the size of the destination is larger than the allowed limit. For the SnapMirror update to be

successful, the size of the destination volume must be reduced to less than the allowed limit for that platform.

Volume SnapMirror locks all volume Snapshot copies during the transfer, and so can lock the temporary Snapshot copy created for the clone operation. If a volume SnapMirror transfer starts while a clone operation is in progress, then the Snapshot copy created on the source FlexVol volume for the cloning operation does not get deleted at the end of the clone operation if the volume SnapMirror transfer is still in progress. Also in this case, when the SnapMirror transfer is still in progress , any attempt to start another clone operation that uses a temporary Snapshot copy even after the completion of the first clone operation on the same FlexVol volume might fail. Further clone operations using temporary Snapshot copies on the source FlexVol volume should be attempted only after the SnapMirror transfer is complete.

5.2 QTREE SNAPMIRROR AND SNAPVAULT

Qtree SnapMirror enables replication of the source volume or qtree to a destination qtree. SnapVault® is a product for protecting data against loss and preserving old versions of data. SnapVault replicates data in primary system paths to qtrees on a SnapVault secondary storage system. SnapVault uses the qtree replication engine to perform replication.

Qtree SnapMirror and SnapVault all work at the logical file level. So, these features are not aware of the fact that the logical files, which are clones, share the same physical blocks. Hence all the clone files and LUNs are transferred to the destination as different physical files and LUNs and are stored on the destination as different files and LUNs. So, no block sharing exists on the destination as all clones are stored as separate files and they occupy space for their own data blocks as shown below.

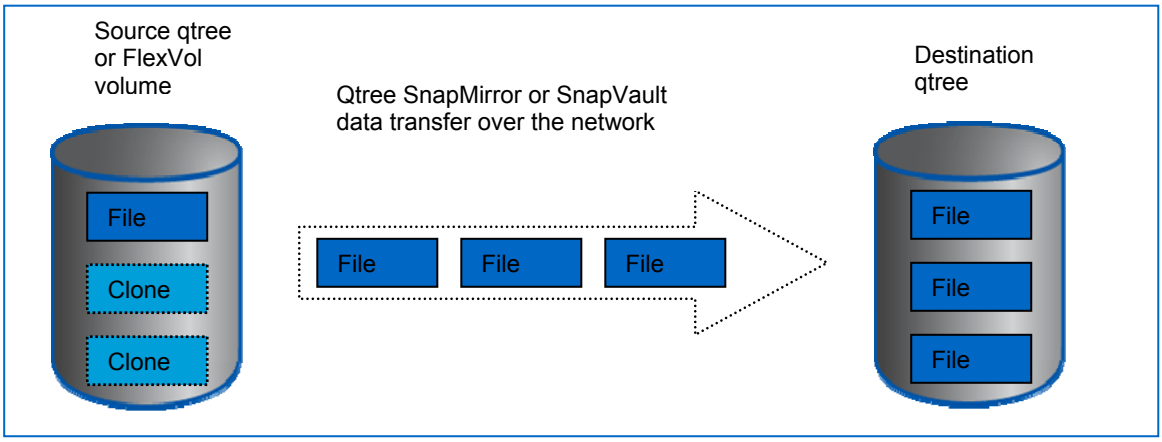


Figure 4) Qtree SnapMirror transfer of FlexClone instance of file or LUN.

The destination size needs to be equal to the sum of the sizes of all logical files and LUNs present on the source. One way to get an estimate of the required size of the destination volume is to run the `df` command with the volume name as argument to figure out the used and saved space. The size of destination must be equal to or greater than the sum of used and save space. Running deduplication on the destination after the SnapMirror transfer is complete will reduce the amount of used space on the destination to be almost the same as the source. `

5.3 NDMP AND DUMP

Dump and NDMP also work at the logical level, just like qtree SnapMirror. So, when you back up a FlexVol volume or qtree, each clone will be backed up as an individual file. When restoring back, they are restored as individual files.

If dump is triggered while a clone operation is in progress, the dump Snapshot copy contains a partially cloned file. The file appears to dump as a partial file and dump is capable of managing such files.

5.4 SYNCHRONOUS SNAPMIRROR

A FlexVol volume that contains a FlexClone instance of a file or LUN that was created using the clone command should not be used as the source of a synchronous SnapMirror relationship. Data ONTAP does not prevent you from establishing a synchronous SnapMirror relationship using a FlexVol volume that has a FlexClone instance of a file or a LUN as the source. However, this is not a supported configuration and so should not be attempted.

5.5 SNAPSHOT

All regular Snapshot operations are supported on a FlexVol volume that has FlexClone instances of files or LUNs in it. If a Snapshot copy is created while a clone operation in progress, the partially cloned file will get locked in the Snapshot copy. Cloning process will complete and a full FlexClone instance of the file will be available for use after the cloning process is complete. The partially cloned file that is locked in the Snapshot copy can be identified by the fact that it has all its permissions set to 0 when the volume is mounted using NFS. An example is shown below:

We have a file called test_file in a FlexVol volume. We now start to create a clone of the file using the clone start command. While the clone operation is in progress, we create a Snapshot copy named testsnap on the storage system. A short while after we create the Snapshot copy, the cloning operation completes. Both the source file and clone are ready to be used by the client as shown below:

```
netapph1%ls -lt

total 21012752

-rw-rw-rw-  1 root root 10737418244 Oct  6  2008 clone_test_file

-rw-rw-rw-  1 root root 10737418244 Oct  6 15:53 test_file
```

However, if we look under the Snapshot directory on the client, we see a partially cloned file with all permissions set to 0. For details on accessing Snapshot directory refer to the Data ONTAP data protection online backup and recovery guide on the NOW site at http://now.netapp.com.

```
netapph1%cd .snapshot

netapph1%cd testsnap

netapph1%ls -lt

total 21012752

----------  1 root root 10737418244 Nov  3 15:45 clone_test_file

-rw-rw-rw-  1 root root 10737418244 Nov  3 15:45 test_file
```

The partial file shows up in the Snapshot copy because the Snapshot copy was created while the clone operation was in progress. Having this partially cloned file in the Snapshot copy has no impact on either the source file or the clone files that are present in the active file system.

FlexClone cannot be used to create a clone of a file or LUN that is present only in a Snapshot copy but not present in the active file system. You can bring the file into the active file system by using SnapRestore on that single file and then create a FlexClone instance of the file.

5.6 SINGLE FILE SNAPRESTORE

Single file SnapRestore® cannot be run in parallel on the same FlexVol volume in which a clone operation is in progress.

5.7 VOLUME OPERATIONS

5.7.1 Volume SnapRestore

Volume SnapRestore cannot be initiated on a FlexVol volume that has a FlexClone instance of a file or LUN operation in progress inside it.

5.7.2 Vol Copy

The source FlexVol volume for a vol copy operation can have FlexClone files or LUNs in it. Once the vol copy operation completes, the destination FlexVol volume will also have the attribute “sis” attached to it and will show up in the output of the vol status command. The file or LUN and its clones on the destination FlexVol volume will share the data blocks just like they did on the source FlexVol volume.

If a vol copy transfer starts while a clone operation is in progress on the source FlexVol volume, then the Snapshot copy created for the clone operation does not get deleted at the end of the clone operation if vol copy is still in progress. Also in this case, any attempt to start another clone operation using a temporary Snapshot copy, after the completion of the first clone operation on the same FlexVol volume while the copy operation is still in progress, might fail. Further clone operations using temporary Snapshot copy on the source FlexVol volume should be attempted only after the copy operation is complete.

5.7.3 Vol Autosize

A FlexClone volume of a file or LUN will occupy some space on the physical storage for storing the metadata. If the FlexVol volume that has vol autosize configured on it and has a creation of a FlexClone instance of a file or LUN in progress on it and the FlexVol volume runs out of space in the middle of creating the metadata required for the clone, the autosize operation will not be activated. So, the size of the FlexVol volume will not be increased by autosize in this case even though the FlexVol volume is full. The clone operation will fail.

When enabling vol autosize on a FlexVol volume that has a FlexClone instance of a file or LUN in it, the maximum size limit for autosize operation must be less than the maximum allowed volume size for that platform. If the limit is more than the maximum allowed size, autosize will not be enabled and an error will be generated.

5.7.4 Vol Clone

A FlexClone volume can be created using a FlexVol volume that has a FlexClone instance of a file or LUN in it as its parent. The clone volume will contain both the file or LUN and its clone. The source and clone files or LUNs that are present in the clone volume will continue to share blocks just like they share blocks in the parent volume. In fact, the file or LUN and its clones that are present in the parent volume and the file or LUN and its clones that are present in the clone volume, all share the same underlying physical data blocks, thus minimizing physical disk space usage. If the clone volume is split from its parent, then the file or LUN and its clones will not share blocks in the child FlexVol volume that was split and so exist as separate files or LUNs. So, the amount of used space in the child FlexVol volume that was split will be greater than what the used space was before the split operation.

FlexClone instance of a file or LUN cannot be created on a FlexClone volume that is currently being split from its parent. Once the split operation is completed, then the FlexClone instance of a file or LUN can then be created on the FlexVol volume that was split.

5.8 MULTISTORE

MultiStore® is a feature of Data ONTAP that enables you to partition the storage and network resources of a single storage system so that it appears as multiple storage systems on the network. Each storage system created as a result of the partitioning is called a vFiler™ unit. For details, see the MultiStore management guide on the NOW site at http://now.netapp.com.

In Data ONTAP 7.3.3 and later, the clone command will work in any vfier context. For more details on using the clone command in a vfiler context refer to the “About FlexClone files and FlexClone LUNs” section of the Data ONTAP storage management guide on the NOW site at http://now.netapp.com/.

In Data ONTAP 7.3.2 and earlier, the clone command does not work in a vFiler context. So, it can only be used in the default vFiler instance, which is vfiler0.

5.9 DEDUPLICATION

A FlexClone instance of a file or a LUN can be created In a FlexVol volume that has deduplication enabled on it. Deduplication shares the physical blocks among different logical files or LUNs. So when creating a FlexClone instance of a file or LUN on a volume that had deduplication already run on it, you might hit the maximum block sharing limit of 255 before you create 255 FlexClone files or LUNs, as the physical block might already be shared during the deduplication process.

The –l flag of the `clone` command enables change logging. This change log information is used by deduplication. For more details about deduplication, change logging, and the `sis` command, refer to NetApp technical report TR-3505.

Enabling change logging might slow down the clone creation process. So change logging is recommended to be enabled only if there is a huge amount of data overwrites to the source file or LUN and you want to share these new data blocks during the deduplication operation.

FlexClone operation cannot be performed on FlexVol volume that has a sis undo operation currently running on it.

5.10 REBOOT

If the storage appliance is rebooted while a creation of a FlexClone instance of a file or LUN is in progress, the FlexClone operation is automatically restarted after the storage system boots up.

5.11 CLUSTER FAILOVER

In case of an HA pair, If a takeover happens and there is a creation of a FlexClone instance of a file or LUN running on the system that is being taken over, the clone operation is terminated prior to the takeover and then automatically restarted after the takeover is complete. Similarly, any clone operations in progress are terminated and automatically restarted after a giveback.

5.12 FLEXSHARE

FlexShare® software is a powerful quality-of-service tool for Data ONTAP storage systems. It lets you assign individual priorities to multiple application workloads consolidated on a single system. The workload generated by creating or deleting a FlexClone instance of a file or LUN is treated as system workload by FlexShare. FlexShare can be used to set a priority to this workload generated by the FlexClone and so its impact on the storage system can be varied according to the priority set for system operations in FlexShare.

5.13 FILE FOLDING

File folding and creation of FlexClone instance of a file or LUN cannot run in parallel on the same FlexVol volume.

5.14 SNAPLOCK

FlexClone at the file and LUN level is not supported on a SnapLock® volume.

6 PERFORMANCE

FlexClone at file and LUN levels is designed to be very efficient as it leverages the internal structure and flexibility of WAFL. In most cases, the storage system performance will not be impacted due to the FlexClone operation. However, under heavy load and for certain types of workloads that are write intensive on the clone or its source and are highly random in nature, FlexClone at file and LUN level might cause some performance impact on the storage system performance for writes to source or its clone. This is because the storage system has to adjust the metadata for the shared blocks being overwritten.

7 EXAMPLE USE CASE

For the purpose of illustrating the ease and efficiency of using FlexClone at file and LUN level, this section walks through a hypothetical use case that involves the complete lifecycle of FlexClone at the file level.

Let’s say there is a company that specializes in developing Image processing software for huge size images such as satellite and space images. Its researchers are continuously working on creating newer and better image processing algorithms for image enhancements that are focused on specific parts of the images that are bad or not clear enough. Since each researcher has a separate algorithm to test on the images, each of them needs a private copy of these huge image files. Since the enhancement algorithms are applied to specific and small parts of the images that are not clear, the changes in the underlying data for these

different images are small compared to the image size. Having so many copies of these huge image files eats up a lot of space, and since most of the image files are similar, storage space is wasted for storing redundant data.

This is where the storage efficiency of FlexClone at file level is put to best use. For illustration purposes let's assume that that the file is approximately 10GB in size.

Also let's assume for illustration purposes that the same volume is mounted to a Linux® client using NFS and also is mapped to a Windows® client using CIFS. This is possible because of the multiprotocol support offered by NetApp storage systems.

Since there is a 10GB file in the volume, here is what the `df` command reports on the storage appliance:

```
netapp01>
netapp01> vol status testvol
         Volume State           Status            Options
        testvol online          raid_dp, flex
                Containing aggregate: 'testaggr'
netapp01>


netapp01> df testvol
Filesystem              kbytes       used      avail capacity  Mounted on
/vol/testvol/         83886080   10509416   73376664      13%  /vol/testvol/
/vol/testvol/.snapshot   20971520          0   20971520       0%
/vol/testvol/.snapshot
netapp01>
```

The `df` command shows that at volume has approximately 10GB of used space, which is consistent with the fact that we have a 10GB file.

Since the volume is mounted on the Linux client using NFS, here is the output of the `ls, df,` and `du` commands from the client:

```
netapph1%ls -lt
total 10506376
-rw-rw-rw-  1 root root 10737418244 Oct  6 15:53 test_file
netapph1%
netapph1%df .
Filesystem           1K-blocks      Used Available Use% Mounted on
netapp01:/vol/testvol
                      83886080  10509728  73376352  13% /tmp/FlexClone
netapph1%.
netapph1%du -B 1K .
4       ./.snapshot
10506384        .
```

The above commands show that the volume has one file of about 10GB. Note that `df` and `du` commands show approximately the same amount of used space.

Here is a screen capture from the Windows box that has the volume mapped using CIFS:

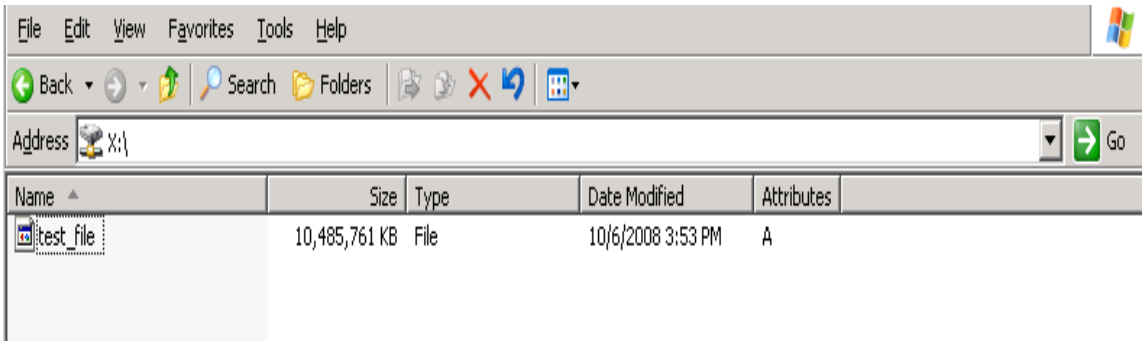


Figure 5) Volume mapped to the client using CIFS

Here are the properties of the share:

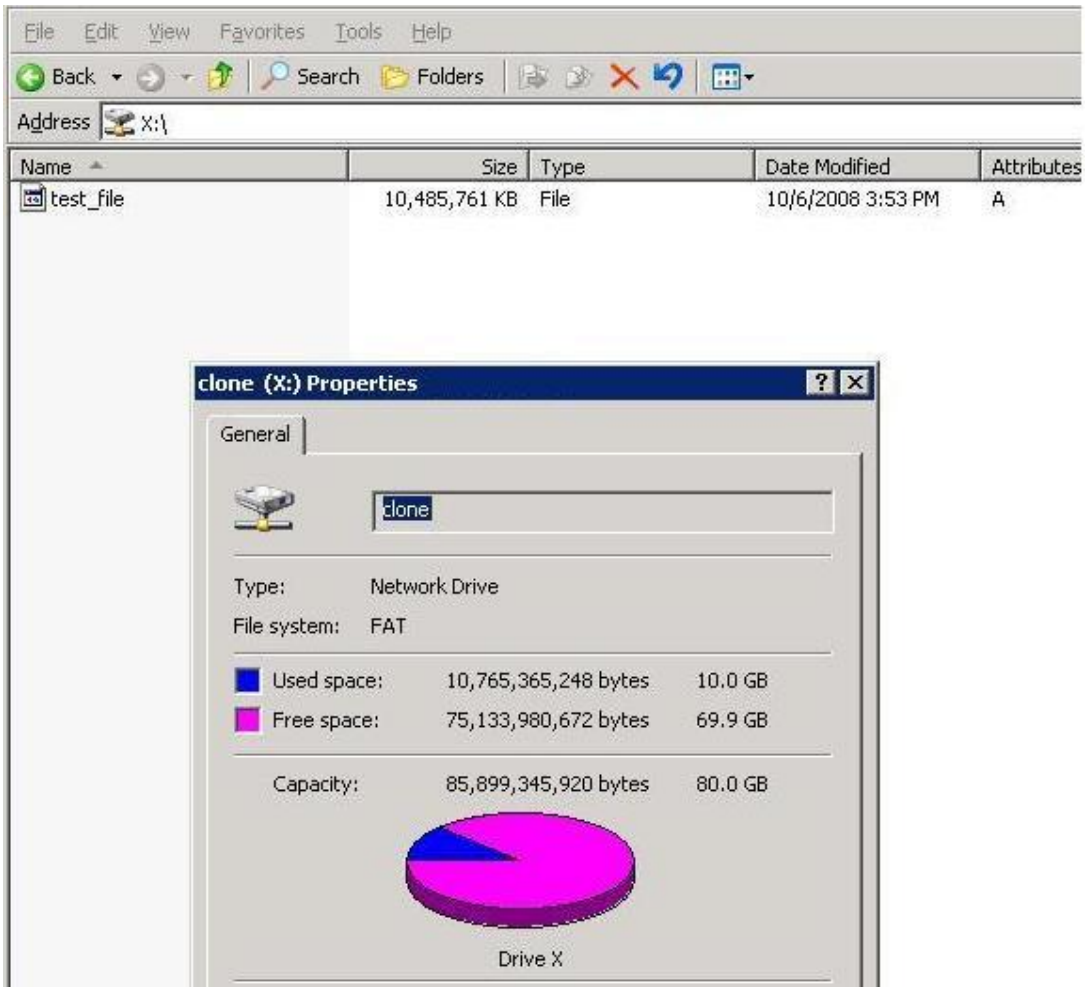


Figure 6) Properties of the CIFS share

As you can see the file size and used space show up as about 10GB, which is what is expected.

Now, suppose two researchers want to work on the same data in this file but would like to experiment with two different algorithms. So, instead of creating two copies of the file and using a total of 20GB in storage space, the storage admin decides to use the new FlexClone enhancements to create a clone of the file.

Here is what the admin does to create the clone:

Start the cloning process using the clone start command:

```
netapp01>

netapp01> clone start /vol/testvol/test file  /vol/testvol/clone test file

Clone operation started successfully. ID: 538.

netapp01> clone status


ID: 538

Source: /vol/testvol/test_file

Destination: /vol/testvol/clone_test_file

Block ranges:

State: running (49% done)

Total blocks: 2621441

Blocks copied: 0

Type: file

netapp01> Mon Oct  6 16:23:29 EDT [netapp01: wafl.snap.delete:info]: Snapshot
copy dense_clone.0.60774330-9572-11dd-9ffe-00a098060974 on volume testvol NetApp
was deleted by the Data ONTAP function dense_clone_delete_snapshot. The unique
ID for this Snapshot copy is (2, 49).

Mon Oct  6 16:23:29 EDT [netapp01: dense.clone.finish:info]: Clone operation on
file '/vol/testvol/clone_test_file' completed successfully. The clone operation
ID was 538.

netapp01>
```

As you see, the clone start command outputs the ID of the clone operation as soon as the cloning process begins. After the completion of the cloning process, the temporary Snapshot copy that was created is deleted and a message is printed indicating the successful completion of the clone. So the storage system now has a 10GB file and one clone of the file. Also notice that the Blocks Copied field in the output of the clone status command has a value of 0. This is because the file and clone share data blocks and there is no physical block copy involved.

Here is what the `df` command shows on the storage system:

```
netapp01> df testvol

Filesystem              kbytes       used      avail capacity  Mounted on

/vol/testvol/         83886080   10538852   73347228      13%  /vol/testvol/

/vol/testvol/.snapshot   20971520          0   20971520       0%
/vol/testvol/.snapshot

netapp01>
```

If you notice the difference between the used space in the output of the `df` command and the used space from the `df` command we issued above prior to starting the clone process, there is only a slightly higher amount of used space now even though we have two 10GB logical files. This is because the file and the clone share physical data blocks. Only extra space occupied by the clone is for storing its metadata.

The clone that was just created must show up to the NFS and CIFS clients as a regular file. Let's go to the clients and figure out what they see.

On the Linux host, the `ls` command lists two files each of size approximately 10GB:

```
netapph1%ls -lt
```

```
total 21012752
-rw-rw-rw-  1 root root 10737418244 Oct  6  2008 clone_test_file
-rw-rw-rw-  1 root root 10737418244 Oct  6 15:53 test_file
netapph1%
```

However, the `df` command on the client shows that the used space is only slightly more than the last time we issued a `df` before cloning:

```
netapph1%df
Filesystem           1K-blocks      Used Available Use% Mounted on
netapp01:/vol/testvol
                      83886080  10538848  73347232  13% /tmp/FlexClone
netapph1%
```

The used space is slightly more than 10GB even though we have two files of size 10GB each. Again, this is because the file and the clone share physical data blocks on the disk.

Here is the output of the `du` command on the Linux host:

```
netapph1%du -B 1K .
4       ./.snapshot
21012760        .
netapph1%
```

As you can see, `du` reports approximately 20GB of used space. This is because there are two files of size 10GB each even though on the storage system, they share data blocks.

The difference in used space as reported by `du` and `df` is an indication that the volume might have a FlexClone instance of a file or LUN in it.

The difference between the used space as reported by `df` and `du` will give a rough estimate of the size of the shared data. So in this example amount of shared data is:

21,012,760 KB – 10,538,848 KB = 10,538,848 KB

So, we have approximately 10GB of shared data, which is true because we have a clone of a 10GB file and the clone and its source share the same data blocks.

Now, let's see what the Windows client with the mapped CIFS share shows:

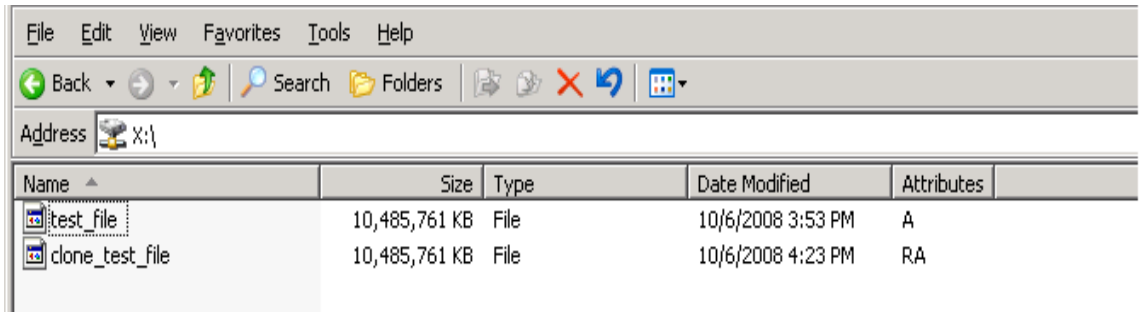


Figure 7) Files as seen from the client

We see two files of size 10GB, just as we saw on the Linux host.

Here are the properties of the share:

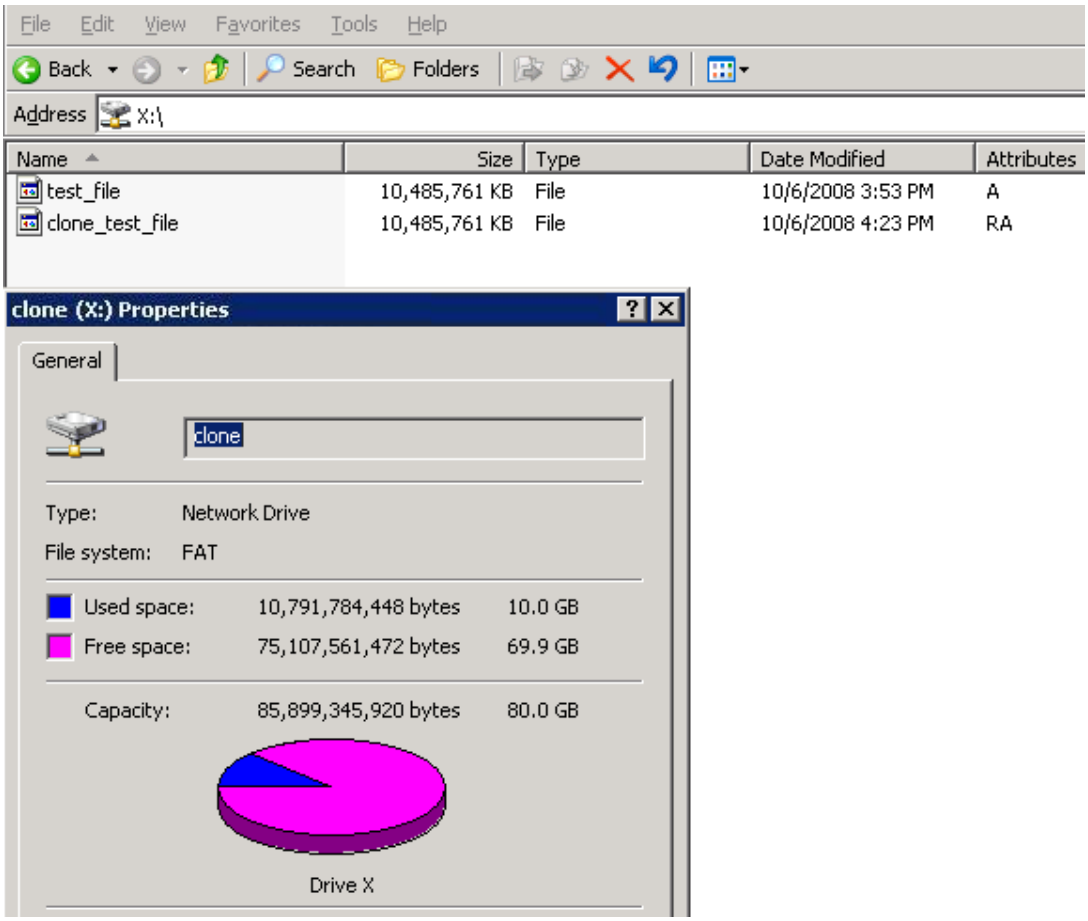


Figure 8) Properties of the share

We see that the total used space on the CIFS share is 10GB even though we have two files each of size 10GB in the share. Again, this is because the file and the clone share physical data blocks on the disk. The amount of shared data can be calculated by subtracting the used space as shown in share properties from the sum of the sizes of the files.

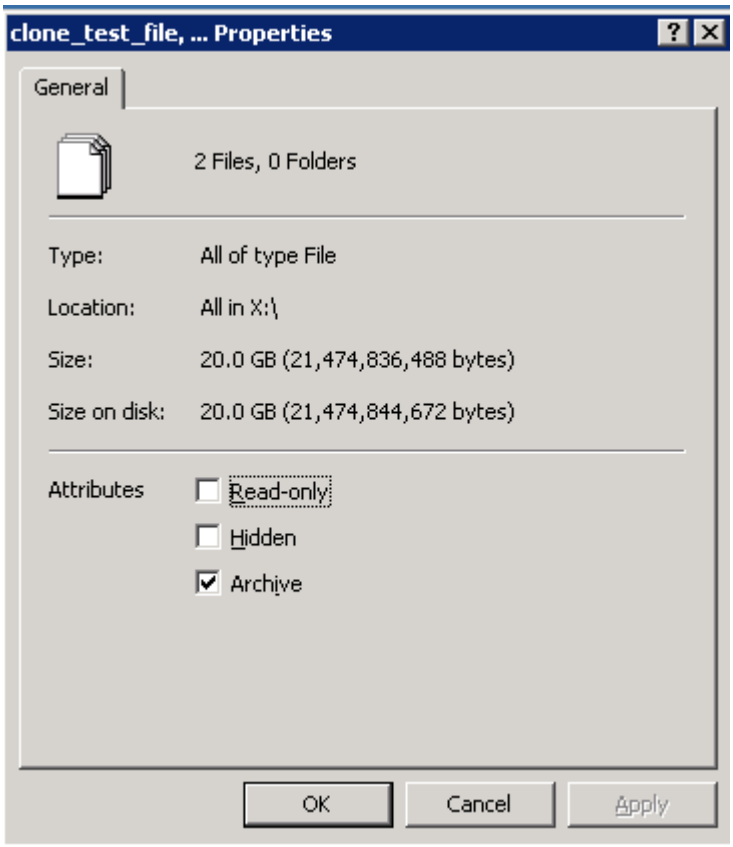


Figure 9) Properties of all the files

So, size of the shared data = 20GB – 10GB = 10GB, which is expected.

Now, both the researchers can start working on their own individual files. Both the file and clone show up in the UNIX or Windows client as regular files. All file operations can be performed on both the clone and its source, including write, read, append, or delete. As data in either the source or the clone gets overwritten, more space on the disk is occupied since the source file and the clone file no longer share the new data blocks that are written.

The "`sis`" attribute now appears in the output of the vol status command since the FlexVol volume now has a FlexClone instance of a file:

```
netapp01>
netapp01> vol status testvol
         Volume State           Status            Options
        testvol online          raid_dp, flex
                                sis
                Containing aggregate: 'testaggr'
netapp01>
```

Now, suppose another researcher comes up with a new algorithm and needs a copy of the file to work with. So, we create another clone as follows:

```
netapp01> clone start /vol/testvol/test_file  /vol/testvol/clone_test_file_2
Clone operation started successfully. ID: 539.
netapp01> Mon Oct  6 16:56:21 EDT [netapp01: wafl.snap.delete:info]: Snapshot
copy dense_clone.0.60774330-9572-11dd-9ffe-00a098060974 on volume testvol NetApp
```

```
was deleted by the Data ONTAP function dense_clone_delete_snapshot. The unique
ID for this Snapshot copy is (3, 80).

Mon Oct  6 16:56:21 EDT [netapp01: dense.clone.finish:info]: Clone operation on
file '/vol/testvol/clone_test_file_2' completed successfully. The clone
operation ID was 539.


netapp01>
```

Here is the output of the `df` command on the storage system:

```
netapp01> df testvol

Filesystem               kbytes       used      avail capacity  Mounted
on

/vol/testvol/          83886080   10539472   73346608 13%  /vol/testvol/

/vol/testvol/.snapshot   20971520          0   20971520       0%
/vol/testvol/.snapshot

netapp01>
```

Notice that the used space is only slightly more than previous used space before creating the second clone. Again, this is because the file and the clone share physical data blocks. Only extra space occupied by the clone is for storing its metadata.

Now the Linux and Windows clients see three files. Here is the output from the Linux client:

```
netapph1%

netapph1%df .

Filesystem           1K-blocks      Used Available Use% Mounted on

netapp01:/vol/testvol

                      83886080  10559456  73326624  13% /tmp/FlexClone

netapph1%ls -lt

total 31519128

-rw-rw-rw-  1 root root 10737418244 Oct  6  2008 clone_test_file_2

-rw-rw-rw-  1 root root 10737418244 Oct  6 16:23 clone_test_file

-rw-rw-rw-  1 root root 10737418244 Oct  6 15:53 test_file

.

netapph1%du -B 1K .

4       ./.snapshot

31519136        .

netapph1%
```

Here is the screen capture from the Windows client:

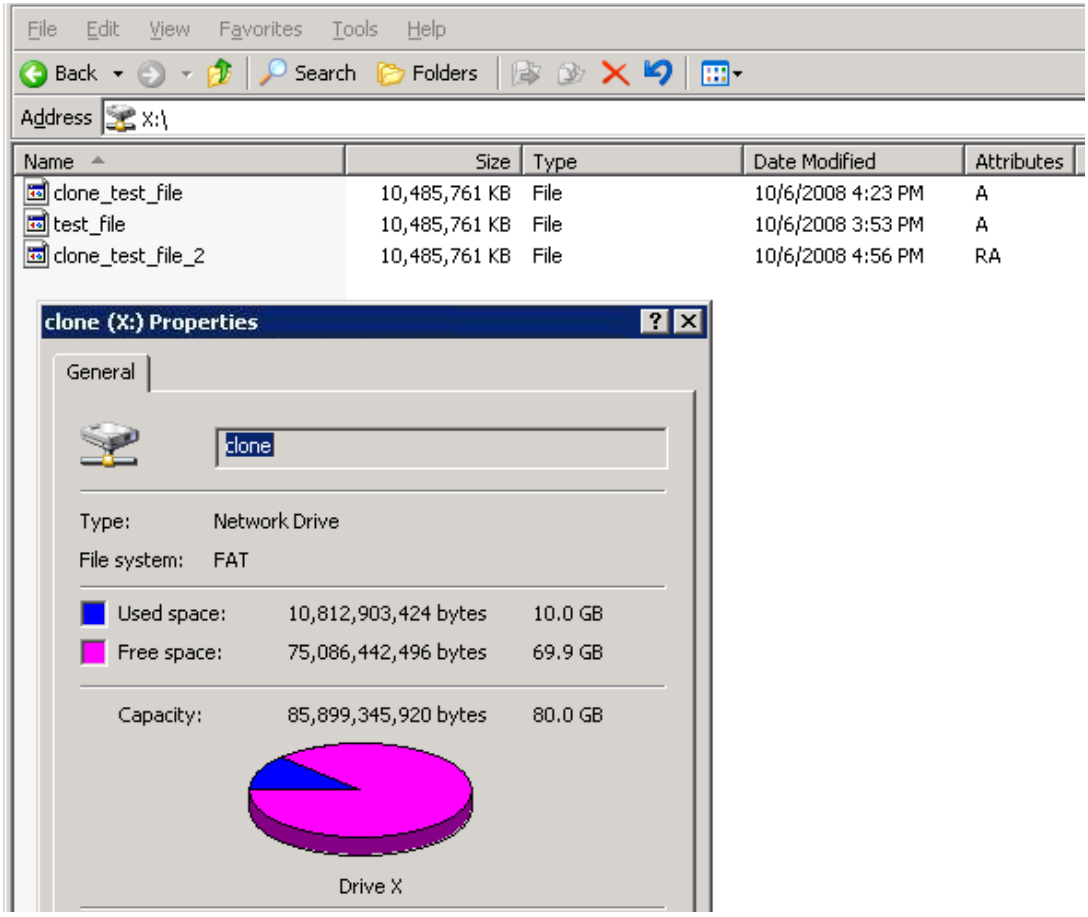


Figure 10) Properties of the share

Notice that both the Linux and Windows client show a total of three files of size 10GB each but show a used space of approximately 10GB. This is because the source and clone files share data blocks.

Now suppose the company has such a huge success with its research efforts that they now have a lot of new image processing algorithms (say 300 in this case) that they want to test out those new algorithms on different copies of the same file. Since the storage admin has the ability to clone files, the admin uses it to create 300 clones of the same file. Since it's burdensome to issue 300 commands by hand, the admin decides to use a script to create the clones. Also the admin is aware of the fact that a file can have a maximum of 255 clone files and after that all clone files will be physical copies of the source that occupy space equal to the source file on the storage system So, the admin writes a script that uses ssh to issue the `clone` command to the storage system to create a total of 255 clones, including the two clones that already exist.

The script is running and it starts creating clones on the storage system. The script issues a bunch of `clone start` commands but suddenly the admin notices that some of the recent commands give the following error message:

```
clone start: Maximum concurrent clone operations already running.
```

Then the admin realizes that there can be a maximum of 16 clone operations that can be active in parallel on a single FlexVol volume. So, the admin modifies the script to add the necessary delays to avoid reaching this limit. The script creates 255 clones.

Here is what the df command reports on the storage system:

```
netapp01> df testvol
Filesystem              kbytes       used      avail capacity  Mounted on
/vol/testvol/         83886080   15776564   68109516      19%  /vol/testvol/
```

```
/vol/testvol/.snapshot    20971520       35056    20936464        0%
/vol/testvol/.snapshot
```

```
netapp01>
```

The Windows client sees 256 files of size 10GB each:

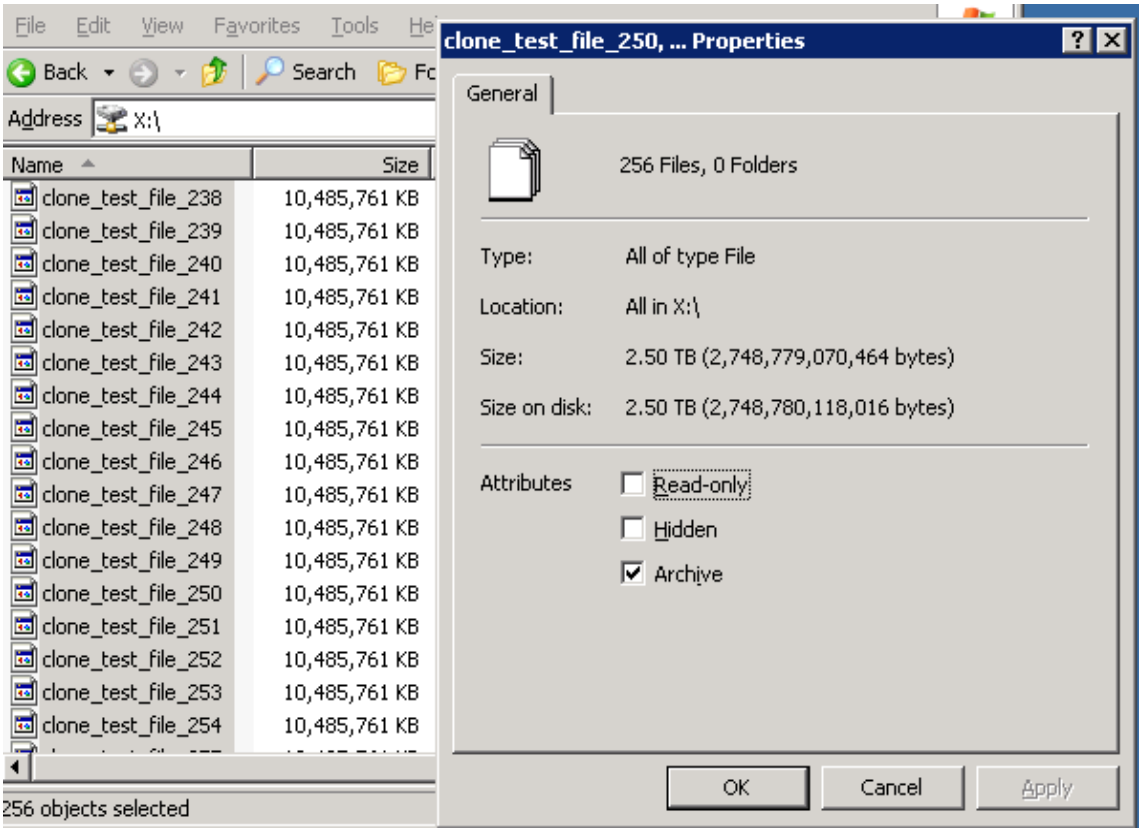


Figure 11) Properties of all the files

Notice that about 15GB of space is used in the volume, as reported by `df` on the storage system even though from a client perspective we have 255 files of size 10GB each. This is because the source and clone files share data blocks.

At this point creating any more clone files of the source file will result in a physical block copy. So the storage admin is careful and wants to closely monitor the space utilization. The admin creates the 256th clone using the clone CLI:

```
netapp01> clone start /vol/testvol/test_file  /vol/testvol/clone_test_file_256
Clone operation started successfully. ID: 802.
netapp01> clone status

ID: 802
Source: /vol/testvol/test_file
Destination: /vol/testvol/clone_test_file_256
Block ranges:
State: running (2% done)
Total blocks: 2621441
Blocks copied: 76500
Type: file
```

```
netapp01>
```

Notice that the clone status command now reports a value of greater than zero for “Blocks copied”. This is because the 255 maximum block sharing limit is reached for all data blocks of the source file and so all physical blocks are now being copied to create the new clone. The blocks copied field now shows the number of blocks being copied. Note that these are 4KB disk blocks. At the end of the cloning process the clone status command reports total blocks and blocks copied as the same, which means the source file was physically copied to create the clone:

```
netapp01> clone status


ID: 802
Source: /vol/testvol/test_file
Destination: /vol/testvol/clone_test_file_256
Block ranges:
State: running (99% done)
Total blocks: 2621441
Blocks copied: 2621441
Type: file
netapp01>
```

This indicates that all the physical data blocks that were present in the source file were copied to create the clone. Once the clone operation is completed, the `clone status` command no longer shows the status of this clone operation and so there is no way to know if the clone is a physical copy or not. Only way to monitor is by using clone status command while the clone operation is in progress.

Here is the output of `df` from the storage system:

```
netapp01>
netapp01> df testvol
Filesystem              kbytes       used      avail capacity  Mounted on
/vol/testvol/         83886080   26285840   57600240      31%  /vol/testvol/
/vol/testvol/.snapshot   20971520      35072   20936448       0%
/vol/testvol/.snapshot
netapp01>
```

Notice that the used space in the FlexVol volume is approximately 25GB, while it was 15GB before creating the 256th clone. This is because, as discussed above, the 256th clone is a physical copy of the source file.

The storage admin then follows the recommendations outlined in previous section of this document, to create any further clones to be based on the 256th clone just created so that he can leverage the WAFL block sharing mechanism to save space as follows:

```
netapp01> clone start /vol/testvol/clone_test_file_256
/vol/testvol/clone_test_file_257
Clone operation started successfully. ID: 803.
netapp01> clone status


ID: 803
Source: /vol/testvol/clone_test_file_256
Destination: /vol/testvol/clone_test_file_257
Block ranges:
State: running (52% done)
Total blocks: 2621441
Blocks copied: 0
```

```
Type: file

netapp01>
```

Notice that clone status shows zero blocks copied because this clone is based on the 256th clone, which was a physical copy. This 256th clone can have another 255 clones based on it before there is another physical copy.

Now suppose some other application creates a huge file on the same volume that has the clones and uses up all the available space. Here is what df says:

```
netapp01> df testvol

Filesystem              kbytes       used      avail capacity  Mounted on

/vol/testvol/         83886080   83877112       8968     100%  /vol/testvol/

/vol/testvol/.snapshot   20971520      35072   20936448       0%
/vol/testvol/.snapshot

netapp01>
```

However, the storage admin is unaware of this and tries to create another clone. The operation starts successfully but fails in the middle because there is not enough space in the FlexVol volume for the clone file to even store its metadata:

```
netapp01> clone start /vol/testvol/clone_test_file_256
/vol/testvol/clone_test_file_258

Clone operation started successfully. ID: 804.

netapp01> Fri Oct 24 18:26:44 EDT [netapp01: wafl.snap.delete:info]: Snapshot
copy dense_clone.0.60774330-9572-11dd-9ffe-00a098060974 on volume testvol NetApp
was deleted by the Data ONTAP function dense_clone_delete_snapshot. The unique
ID for this Snapshot copy is (109, 2400).

Fri Oct 24 18:26:44 EDT [netapp01: dense.clone.failed:error]: Clone operation on
file '/vol/testvol/clone_test_file_258' failed with error 'Not enough disk space
for the operation'. The clone operation ID is 804.

netapp01>
```

The clone status command gives the failure reason:

```
netapp01> clone status


ID: 804

Source: /vol/testvol/clone_test_file_256

Destination: /vol/testvol/clone_test_file_258

Block ranges:

State: failed (Not enough disk space for the operation)

Type: file

netapp01>
```

As stated in the previous section, when the clone process fails, it stores a small amount of metadata on the disk to indicate the failure. This metadata can be cleared by using the clone clear command:

```
netapp01> clone clear testvol 804
```

Now the `clone status` command displays no failures because we just cleared the failure related metadata from the status metadata file:

```
netapp01> clone status

netapp01>
```

The admin now goes ahead and removes the huge file that was created by another application without his knowledge. So, the disk utilization is back to normal:

```
netapp01> df testvol
Filesystem              kbytes       used      avail capacity  Mounted on
/vol/testvol/         83886080   26285840   57600240      31%  /vol/testvol/
/vol/testvol/.snapshot   20971520      35072   20936448       0%
/vol/testvol/.snapshot
netapp01>
```

The admin now wants to figure out how much space was saved by using FlexClone instead of creating the physical copies of the files. The admin uses the df –s command for that:

```
netapp01> df -sh testvol
Filesystem               used      saved       %saved
/vol/testvol/             25GB     2560GB          99%
netapp01>
netapp01>
```

Since there are a total of 256 logical files of 10GB each, they would have taken 2560GB of space. Since we used FlexClone, we only have two physical files of 10GB each, and all the other files are clone files, which occupy space only for their metadata. So, the real physical used space is 25GB.

Now that the files are ready for testing by the researchers for use, they start running their algorithms on the files. Suppose that the initial source file is the first to test the algorithm on. Let's assume that the algorithm overwrites about 1GB of data in the file.

As data in the source or clone is overwritten, the blocks that are overwritten will start to occupy space on the disk. So, after testing is complete on the first file, the admin notices that disk utilization has gone up by approximately 1GB:

```
netapp01>
netapp01> df testvol
Filesystem              kbytes       used      avail capacity  Mounted on
/vol/testvol/         83886080   27360592   56525488      33%  /vol/testvol/
/vol/testvol/.snapshot   20971520      32440   20939080       0%
/vol/testvol/.snapshot
netapp01>
netapp01> df -h testvol
Filesystem               total       used      avail capacity  Mounted on
/vol/testvol/             80GB       26GB       53GB      33%  /vol/testvol/
/vol/testvol/.snapshot       20GB       31MB       19GB       0%
/vol/testvol/.snapshot
```

As you can see, the used space, as seen by the output of df command before and after overwriting the first file, has increased from 26285840KB to 27360592KB, which is approximately a 1GB increase. This is because 1GB of data was overwritten in the file.

Now that the testing on the first file is complete, the admin can go ahead and delete the file:

```
netapph1%
netapph1% rm test_file
netapph1%
```

As discussed earlier, deleting the source file or any of its clones will not affect the other clone files in any way, and they continue to share blocks on disk. Here is what the `df` command outputs:

```
netapp01> df testvol
```

```
Filesystem              kbytes       used      avail capacity  Mounted on
/vol/testvol/         83886080   26291404   57594676      31%  /vol/testvol/
/vol/testvol/.snapshot   20971520    1106504   19865016       5%
/vol/testvol/.snapshot

netapp01>

netapp01> df -h testvol
Filesystem               total       used      avail capacity  Mounted on
/vol/testvol/             80GB       25GB       54GB      31%  /vol/testvol/
/vol/testvol/.snapshot       20GB     1080MB       18GB       5%
/vol/testvol/.snapshot

netapp01>
```

Notice that by deleting the 10GB file, we only got about 1GB of space back. That's because the 1GB was the data overwritten in the file in the course of testing. Even though the file was of size 10GB, this 1GB is the data blocks that were not shared. So, once the file was deleted we got the 1GB of space back. When a file or clone that has shared blocks is deleted, the shared blocks continue to be used by the remaining clone files or LUNs. If there are other clone files that share the data blocks, deleting a file or its clone will only return to free space the space used for the metadata of that file or clone and the space used by the data blocks of that file or clone that are not shared. So, not much space is returned to free space until the source file or LUN and all clone files or LUNs of that source are deleted.

Even though the initial source file is deleted, all the clone files still exist and show up to the NFS or CIFS clients as normal files. They also continue to share blocks on disk.

Now, the admin gets word from the researchers that 12 of their algorithms are flawed and so they do not need about 12 of the files that were given to them. So, the admin can go ahead and delete 12 of the clone files. The admin realizes that when creating the clones, that 256th clone was a full physical copy since the block sharing limit was hit, and the admin decides to first delete the 256th clone file, which is a full physical copy, and the 257th clone that is based on it.

So, the admin deletes the 257th clone:

```
netapph1%

netapph1% rm clone_test_file_257

netapph1%
```

However, the used space on the storage system will not change much after deleting the 257th clone since the 257th and 256th clone share data blocks and the 256th clone is still present.

Now, the admin deletes the 256th clone:

```
netapph1%

netapph1% rm clone_test_file_256

netapph1%
```

Now, the used space on the storage system has gone down by about 10GB:

```
netapp01>

netapp01> df testvol
Filesystem              kbytes       used      avail capacity  Mounted on
/vol/testvol/         83886080   15764424   68121656      19%  /vol/testvol/
/vol/testvol/.snapshot   20971520   11639016    9332504      55%
/vol/testvol/.snapshot

netapp01>

netapp01> df -h testvol
Filesystem               total       used      avail capacity  Mounted on
```

```
/vol/testvol/              80GB       15GB       64GB      19%  /vol/testvol/
/vol/testvol/.snapshot       20GB       11GB     9113MB      55%
/vol/testvol/.snapshot
netapp01>
```

This is because both the source file and its clone file are deleted, the data blocks are returned to the free space pool.

Now the admin goes ahead and deletes 10 more files as initially planned:

netapph1% rm clone_test_file_255;rm clone_test_file_254;rm clone_test_file_253; rm clone_test_file_252; rm clone_test_file_251; rm clone_test_file_250; rm clone_test_file_249; rm clone_test_file_248; rm clone_test_file_247; rm clone_test_file_246

The df command reports that used space has not changed much:

```
netapp01> df testvol
Filesystem              kbytes       used      avail capacity  Mounted on
/vol/testvol/         83886080   15558308   68327772      19%  /vol/testvol/
/vol/testvol/.snapshot   20971520   11845644    9125876      56%
/vol/testvol/.snapshot
netapp01>
netapp01> df -h  testvol
Filesystem               total       used      avail capacity  Mounted on
/vol/testvol/              80GB       14GB       65GB      19%  /vol/testvol/
/vol/testvol/.snapshot       20GB       11GB     8911MB      56%
/vol/testvol/.snapshot
netapp01>
```

The used space has not changed much because the remaining clones continue to share the physical data blocks on the disk and continue to reference them.

The above use case example discusses in detail the complete lifecycle of a clone file in the cloning process and the different conditions encountered while using the clone feature. It also highlights the advantages of FlexClone at file level and the usage steps for maximum efficiency.

As discussed in the previous sections of this document, the process to create a FlexClone instance of a LUN is the same as creating a FlexClone instance of a file. Here is the process to create a FlexClone instance of a LUN:

- Make sure the LUN exists:

```
netapp01> lun show /vol/testvol/testlun
        /vol/testvol/testlun        10.0g (10742215680)   (r/w, online, mapped)
netapp01> lun show -v /vol/testvol/testlun
        /vol/testvol/testlun        10.0g (10742215680)   (r/w, online, mapped)
                Serial#: HnSrBoMaK8ZF
                Share: none
                Space Reservation: enabled
                Multiprotocol Type: windows
                Maps: igroup1=1
```

- Create a FlexClone instance of the LUN:

```
netapp01> clone start /vol/testvol/testlun  /vol/testvol/clone_testlun
Clone operation started successfully. ID: 9.
netapp01> Fri Dec 19 12:28:07 GMT [netapp01: wafl.snap.delete:info]: Snapshot
copy dense_clone.0.2e0aa696-cf59-11dd-94a1-00a09803488b on volume testvol NetApp
```

```
was deleted by the Data ONTAP function dense_clone_delete_snapshot. The unique
ID for this Snapshot copy is (1, 7).

Fri Dec 19 12:28:07 GMT [netapp01: dense.clone.finish:info]: Clone operation on
file '/vol/testvol/clone_testlun' completed successfully. The clone operation ID
was 9.

netapp01>
```

8 FLEXCLONE IN VIRTUALIZED ENVIRONMENTS

Virtualized environments provide a perfect use case for FlexClone. A virtual machine (VM) is made up of one or many binary files that are the disks of the virtual machine, and some set of associated files or metadata, which define the configuration of the VM. These virtual disk files can range in size from a few gigabytes to tens of gigabytes depending on their use.

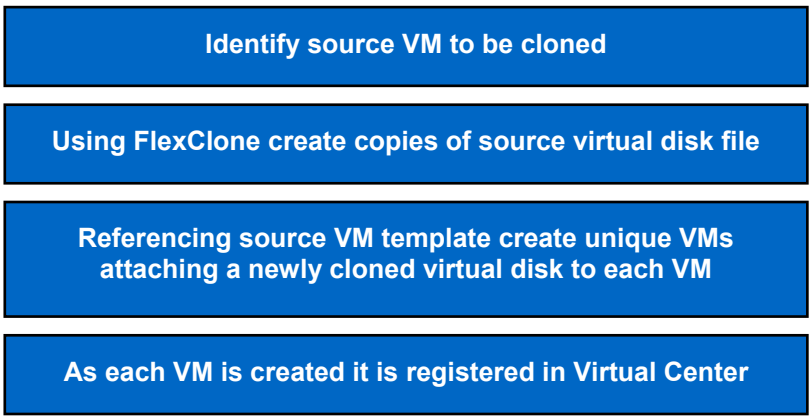
In a virtualized environment new virtual machines can be deployed using an existing VM as a template and cloning a new VM from that template. During this process a copy is made of the template virtual disk file and new configuration files or metadata are generated and associated with the new virtual disk to create a new VM. The copy of the virtual disk file is the most time consuming part of the process and is performed by the physical server hosting the virtual machine.

NetApp FlexClone provides the ability to offload the work of copying the virtual disks from the servers to the storage appliance.

In virtual desktop infrastructure (VDI) environments the ability to quickly clone large numbers of VMs becomes even more attractive. In these environments there are typically hundreds or thousands of VMs as compared to the number of virtual servers you might see in a virtual server environment. Also, in terms of VDI, there are cases where the cloning or deployment and redeployment of desktops provide greater flexibility in a VDI solution if the storage is capable of creating clones very quickly. For example, keeping up with operating system updates by patching a single copy of a VM and then redeploying that VM by cloning it thousands of times is an attractive solution. However, it is not possible to redeploy thousands of VMs in a reasonable frame of time using host-based copy methods. If the storage device does not enable a method of quick redeployment, then this type of process can be impossible to implement. Cloning hundreds or thousands of VMs takes time, for example, six to 12 minutes per VM or five to 10 per hour. That means to redeploy one thousand VMs could take over four days, and during this time the servers suffer the additional workload of traditional copy processes.

FlexClone at the file level eliminates the I/O necessary to clone the VMs enabling the cloning of mass numbers of virtual machines in minutes. However, there is more to creating clones than simply making copies of the virtual disks. For example, in a VMware® environment, VMs need to be registered in the Virtual Center Server and then customized to make them unique entities that can be joined to a Microsoft® Active Directory domain.

Here is an example workflow of the VM cloning process. In this workflow NetApp FlexClone is used to replace the copy operation that would normally be performed by the host when creating multiple clones of VMs running Microsoft Windows as the guest operating system.



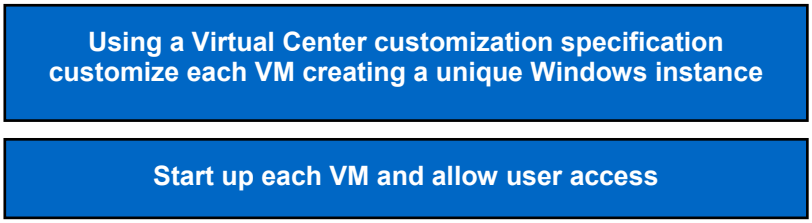


Figure 12) VM cloning process

NetApp has provided a utility to automate the entire process of cloning, registration, customization, and deployment of virtual machines. This utility is called the Rapid Cloning Utility (RCU). The RCU makes use of the native toolsets in the VMware Virtual Infrastructure SDK and the NetApp ONTAPI SDK including FlexClone, and VMware Virtual Center features, such as Guest Customization to efficiently clone and customize virtual machines. To customize VMs the RCU uses existing Guest Customization specifications stored within VMware Virtual Center. Monitoring of cloning activities can be done in the Virtual Infrastructure Client, where actions are displayed as they are performed on the VMs. Information about the cloning process is also collected in the Data ONTAP messages file.

The Rapid Cloning Utility is discussed in NetApp technical report TR-3705.

9 CONCLUSION

FlexClone technology provides a space- and time-efficient way to create and manage multiple copies of data on the storage system. The recent enhancements to FlexClone technology with increased granularity and Snapshot copy independence provide a powerful tool for storage administrators to rapidly deploy multiple copies of data sets while saving storage space.

10 ACKNOWLEDGEMENTS

The author would like to thank all the people who provided the material for this report and the reviewers for their valuable feedback.

11 REFERENCES

11.1 NETAPP TECHNICAL REPORTS AND WHITE PAPERS

- TR-3001: A Storage Networking Appliance
  - www.netapp.com/us/library/technical-reports/tr-3001.html
- TR-3002: File System Design for an NFS File Server Appliance
  - www.netapp.com/us/library/white-papers/wp_3002.html
- TR-3347: A Thorough Introduction to FlexClone Volumes
  - www.netapp.com/us/library/technical-reports/tr-3347.html
- TR-3505: NetApp Deduplication for FAS and V-Series Deployment and Implementation Guide
  - www.netapp.com/us/library/technical-reports/tr-3505.html
- TR-3705: NetApp and VMware View (VDI) Solution Guide
  - www.netapp.com/us/library/technical-reports/tr-3705.html

11.2 DATA ONTAP PRODUCT DOCUMENTATION

- System Administration Guide
  - http://now.netapp.com/NOW/knowledge/docs/ontap/ontap_index.shtml
- Data Protection Online Backup and Recovery Guide
  - http://now.netapp.com/NOW/knowledge/docs/ontap/ontap_index.shtml
- MultiStore Management Guide
  - http://now.netapp.com/NOW/knowledge/docs/ontap/ontap_index.shtml
- Storage Management Guide
  - http://now.netapp.com/NOW/knowledge/docs/ontap/ontap_index.shtml