Alex Osuna
Richard Jooss

Thin Provisioning in an IBM SAN or IP SAN Enterprise Environment

Introduction

This IBM® Redpaper describes how to provision storage efficiently with IBM System Storage™ N series in a Fibre Channel or iSCSI deployment. Storage efficiency has historically been extremely low because of the difficulty provisioning or changing provisioning to adapt to changing needs with classic storage array architectures. System Storage N series with FlexVol™ technology provides an extremely flexible paradigm for provisioning, allowing the user to minimize unused space.

Note: This paper does not discuss thin provisioning using LUN and FlexClone.

Thin provisioning

Thin provisioning presents more storage space to the hosts or servers that are connected to the storage system than is actually available on the storage system. System Storage N series have always had this capability for Fibre Channel. iSCSI provisioning and FlexVol technology further increased this flexibility.

An example of thin provisioning is when a storage system contains 5000 GB of usable storage capacity, but the storage administrator has mapped LUNs of 500 GB each to 15 hosts. In this example, the storage administrator makes 7500 GB of storage space visible to

the hosts even though the storage system has only 5000 GB of usable space (see Figure 1). If all 15 hosts immediately use all 500 GB provisioned to them, there would be a problem. The storage administrator has to monitor the system and add storage as needed.

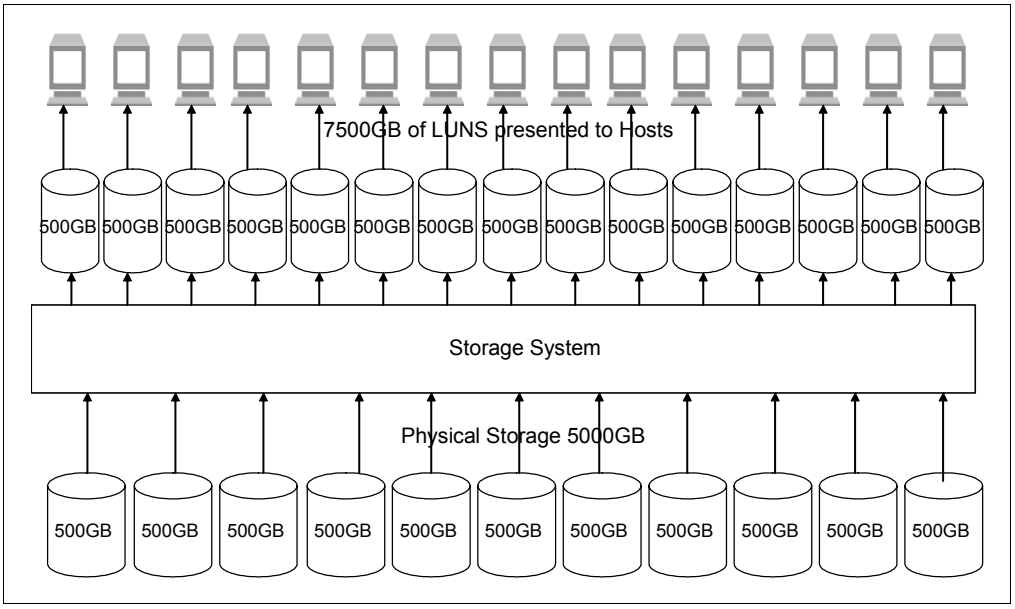

Figure 1 Thin provisioning example

In other areas or industries the practice of thin provisioning is prevalent. For example, a water utility company's infrastructure does not have the capacity to maintain acceptable water pressure if all of its customers turn on their taps simultaneously. The company plans on having only a certain percentage of its customers simultaneously using water. In the banking industry, clearly banks do not have enough cash on hand to allow all of their customers to withdraw all the money in their accounts at one time. They plan that only a small percentage of their customers will be withdrawing money at one particular time. Both of these examples illustrate that thin provisioning is more applicable in larger environments, including the storage world. The larger the storage system and the more users or applications utilizing the storage typically the better the chance to take advantage of thin provisioning.

WAFL: The enabling technology

WAFL® (write anywhere file layout), which can be thought of as the virtualization layer of Data ONTAP®, is the technology that enables thin provisioning. When a LUN is created, it does not dedicate specific blocks out of the System Storage N series volume for the LUN or for Snapshot™ copies of the LUN. Instead, it allocates the blocks from the System Storage N series volume when the data is actually written. This allows the administrator to provision more storage space, as seen from the connected servers, than is physically present in the storage system.

Host versus storage view

There is often confusion over the space usage inside a LUN. This confusion stems from the fact that the vast majority of LUNs have a file system installed on them, and how full the file system might be is not the same as how full the LUN is. Assuming the file system is configured to have access to the complete LUN, the LUN will always be at least as full as the file system. From a storage system perspective, the percentage of a LUN used will always increase and never decrease.

Figure 2 helps show why an individual LUN's usage never decreases. The figure shows a theoretical case of how a file system and a LUN are used as files and are written and deleted from the file system.

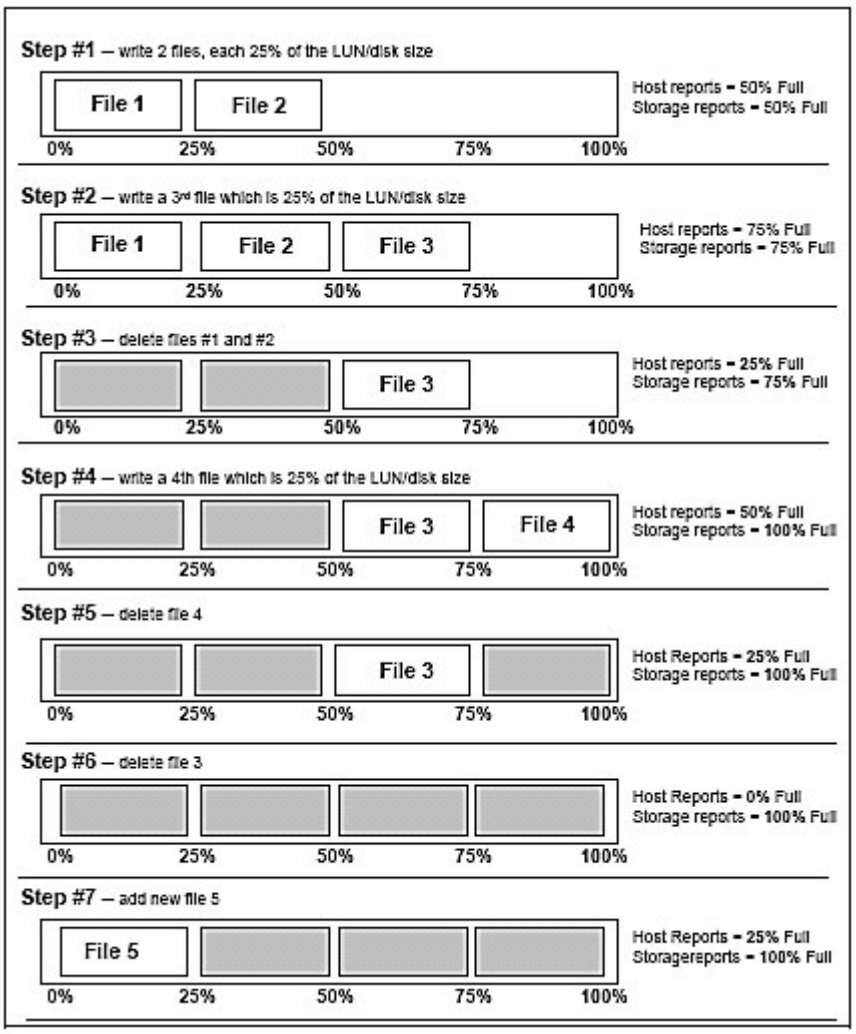

Figure 2 LUN capacity: Host versus storage view

Steps 1 and 2 show, as expected, that the file system and the LUN both report the same percentage used as files are written to the file system. Step 3 is where a difference occurs between what the host or file system reports and what the storage reports. In this step, files 1 and 2 are deleted, meaning that the blocks used by these files are available to the file system. The only file still visible in the file system is file 3. So, the file system reports it is 25% full. However, the storage is still reporting it is 75% full. The reason that the storage shows 75% full is because the storage has no way of knowing that the data written for files 1 and 2 is no longer needed. The space in the graphic is shaded because from the storage perspective those blocks still contain valid data. When a file is deleted the file system does not send any type of erase or delete command to the storage array. It simply changes a pointer in its tables that pointed to the file and makes those blocks available for future use. The fact that the blocks containing the files are not actually erased is what allows *undelete* utilities to recover files after the user has deleted them.

In Step 4, file 4 is shown being written to the last 25% of the LUN. This step depicts how a file system does not necessarily attempt to keep data at the beginning of a LUN. This is also shown in Figure 3, which shows the layout of an NTFS file system that is 75% full. After Step 4, every block in the LUN has been written and is, therefore, full from a storage perspective. As mentioned, the fill percentage of a LUN will never go down and Step 5 through Step 7 show that, regardless of the host activity on the LUN, it will remain 100% full.

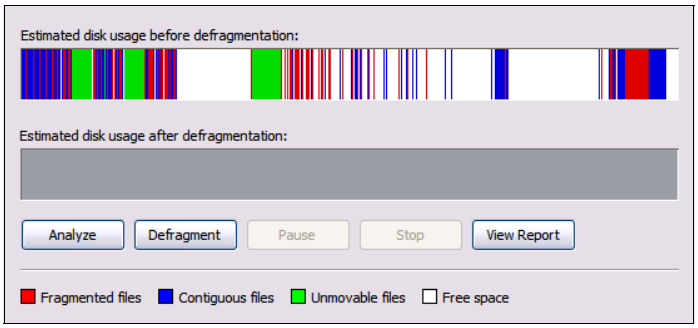

Figure 3 LUN usage

LUN capacity used over time

Figure 4 shows how much of a LUN's capacity is used over time from a storage perspective. The steepness of the curve and how fast the LUN will get to 100% is affected by many parameters, such as the type of file system or application, the size of the files being written to the file system, and the percentage of the file system itself that is being used. The time it takes for a LUN to reach 100% will typically be measured in days, weeks, or months, not seconds or years.

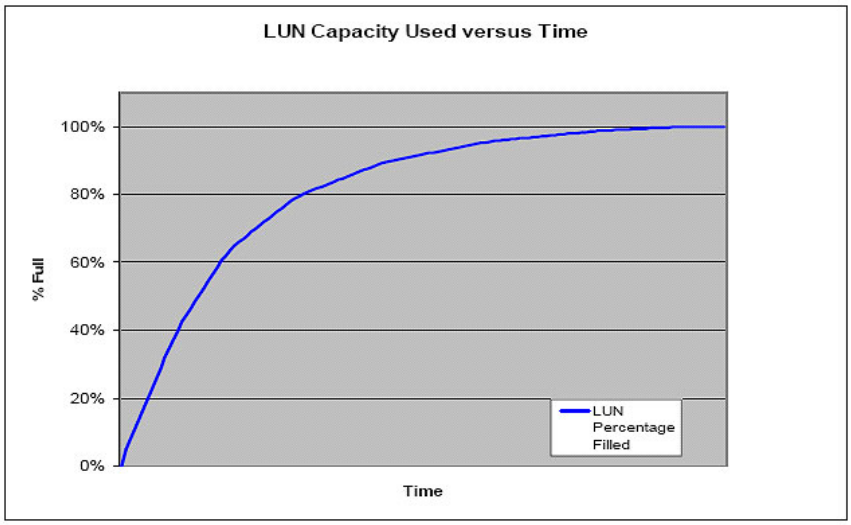

Figure 4 LUN capacity used versus time

Data ONTAP configuration parameters

This section discusses a number of variables or options in Data ONTAP that are important to understand before configuring for thin provisioning.

LUN reservation

LUN reservation (not to be confused with *SCSI2* or *3 logical unit locking reservations*) determines when space for the LUN is reserved or allocated from the volume. With reservations enabled (default) the space is subtracted from the volume total when the LUN is created. For example, if a 20 GB LUN is created in a volume having 80 GB of free space, the free space will go to 60 GB at the time the LUN is created even though no writes have been performed to the LUN. If reservations are disabled, space is first taken out of the volume as writes to the LUN are performed. If the 20 GB LUN was created without LUN space reservation enabled, the free space in the volume would remain at 80 GB and would only go down as the LUN was written to.

Guarantees

With flexible volumes, there is the new concept of space *guarantees*, which allow the user to determine when space is reserved or allocated from the containing aggregate:

- *Volume*: A guarantee of *volume* ensures that the amount of space required by the FlexVol volume is always available from its aggregate. This is the default setting for FlexVol volumes. With the space guarantee set to *volume* the space is subtracted, or reserved, from the volume's available space regardless of whether it is actually used for data storage.

  Example 1 shows the creation of a 20 GB volume. The **`df`** commands show the space usage aggregate before and after the **`vol create`** command shows how the 20 GB is removed from the aggregate as soon as the volume is created, even though no data has actually been written to the volume.

Example 1 Volume creation

```
itsotuc1> df -A -g aggr0
Aggregate              total used  avail capacity
aggr0                  85 GB 0GB   85GB 0%
aggr0/.snapshot        4GB   0GB   4GB  0%

itsotuc1> vol create flex0 aggr0 20g
Creation of volume 'flex0' with size 20g on hosting aggregate 'aggr0' has
completed.

itsotuc1> df -g /vol/flex0
Filesystem             total used  avail capacity Mounted on
/vol/flex0/            16GB   0GB  15GB 0%          /vol/flex0/
/vol/flex0/.snapshot   4GB   0GB   4GB  0%
/vol/flex0/.snapshot
itsotuc1> df -A -g aggr0
Aggregate              total used  avail capacity
aggr0                  85GB  20GB  65GB 23%
aggr0/.snapshot        4GB   0GB   4GB  0%
```

Because the space has already been reserved from the aggregate, write operations to the volume will not cause more space from the aggregate to be used.

- *None*: A FlexVol volume with a guarantee of *none* reserves no space, regardless of the space reservation settings for LUNs in that volume. Write operations to space-reserved LUNs in that volume might fail if the containing aggregate does not have enough available space. Space is first taken from the aggregate when data is actually written to the volume. Example 2 shows how, in contrast to the example with the volume guarantee, the volume creation does not affect the used space in the aggregate. Even LUN creation, which by default has space reservation enabled, does not reserve space out of the volume.

Example 2 Volume creation with guarantee of none

```
itsotuc1> df -A -g aggr0
Aggregate                      total  used   avail  capacity
aggr0                   85GB                 0GB    85GB   0%
aggr0/.snapshot            4GB       0GB    4GB    0%
itsotuc1>
itsotuc1> vol create noneflex -s none aggr0 20g
Creation of volume 'noneflex' with size 20g on hosting aggregate
'aggr0' has completed.
itsotuc1> df -g /vol/noneflex
Filesystem                     total  used availcapacity                Mounted on
/vol/noneflex/                 16GB    0GB 16GB38%                  /vol/noneflex/
/vol/noneflex/.snapshot         4GB    0GB    4GB    0%
/vol/noneflex/.snapshot
itsotuc1> df -A -g aggr0
Aggregate                      total  used availcapacity
aggr0                          85GB   0GB 85GB0%
aggr0/.snapshot                4GB    0GB 4GB 0%
itsotuc1> lun create -s 10g -t windows /vol/noneflex/foo
Mon Nov 24 15:17:28 EST [array1:
lun.vdisk.spaceReservationNotHonored:notice]: Space reservations in
noneflex are not being honored, either because the volume space guarantee
is set to 'none' or the guarantee is currently disabled due to lack of space in
the aggregate.
lun create: created a LUN of size:   10.0g (10742215680)
itsotuc1> df -A -g aggr0
Aggregate                      total   used availcapacity
aggr0                          85GB   0GB 85GB0%
aggr0/.snapshot                4GB    0GB 4GB 0%
```

- *File*: The aggregate guarantees that space is always available for overwrites to space-reserved LUNs. Fractional reserve, a volume level option discussed later in this paper, is set to 100% and is not adjustable with this type of space reservation. The *file* guarantee is basically the same as the *none* guarantee with the exception that space reservations for LUNs and space-reserved files are honored.

Example 3 looks the same as the previous example under the *none* reservation except that the LUN creation takes space from the aggregate because it is a space-reserved object by default. Because the space reservation is honored, the **`lun create`** command also does not issue the warning shown in the Figure 2.

Example 3 LUN create with file guarantee

```
itsotuc1> df -A -g aggr0
Aggregate                total       used      availcapacity
aggr0                     85GB        0GB       85GB       0%
aggr0/.snapshot            4GB        0GB        4GB       0%
itsotuc1>
itsotuc1> vol create noneflex -s none aggr0 20g
Creation of volume 'noneflex' with size 20g on hosting aggregate
'aggr0' has completed.
cnrl1>
itsotuc1> df -g /vol/noneflex
Filesystem               total       used      avail capacity  Mounted on
/vol/noneflex/            16GB        0GB       16GB      38%
/vol/noneflex/
/vol/noneflex/.snapshot    4GB        0GB        4GB           0%
/vol/noneflex/.snapshot
itsotuc1>
itsotuc1> df -A -g aggr0
Aggregate                total       used      avail capacity
aggr0                     85GB        0GB       85GB       0%
aggr0/.snapshot            4GB        0GB        4GB       0%
itsotuc1>
itsotuc1> lun create -s 10g -t windows /vol/noneflex/foo
lun create: created a LUN of size:   10.0g (10742215680)
itsotuc1> df -A -g aggr0
Aggregate                total       used      avail capacity
aggr0                     85GB       10GB       75GB       0%
aggr0/.snapshot            4GB        0GB        4GB       0%
```

Fractional reserve

Fractional reserve is a volume option that determines how much space Data ONTAP will reserve for Snapshot overwrite data for LUNs and space-reserved files. The default value is 100%. Data ONTAP removes or reserves this space from the volume as soon as the first Snapshot copy is created. For example, as shown in Figure 5 on the left-hand side, a 100 GB volume is shown with two 20 GB LUNs. Assuming the LUNs are full and a Snapshot copy is created, Data ONTAP by default will reserve 40 GB (2 x 20 GB) of space in the volume to assure that there is always enough space for both the LUNs and all the Snapshot data, even if the LUNs are completely overwritten.

This example is shown in the middle of Figure 5. As depicted on the right-hand side, if `fractional_reserve` is set to 60% when the Snapshot is created, instead of reserving 40 GB in the volume, Data ONTAP will reserve only 24 GB (60% * [2 x 20 GB]).

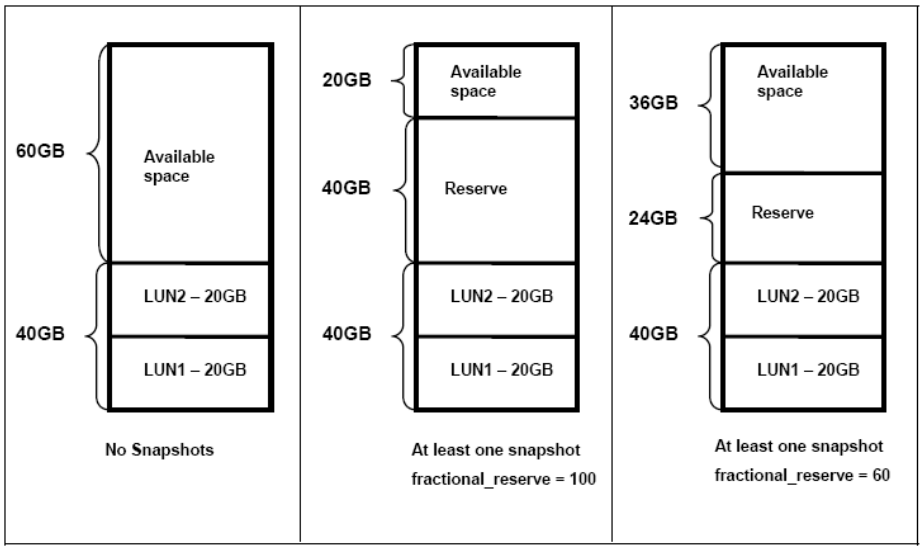

Figure 5 Fractional reserve

The amount of space reserved can be seen using the **`-r`** option of the **`df`** command. This reserve area is used only when Data ONTAP reports that the volume is full. Until the volume is full, space for Snapshot overwrites are taken from the volume and only when the volume is 100% full will this reserved space be used.

Snap reserve

The snap reserve variable is set at the volume level and is set as a percentage of the volume. Data ONTAP removes the defined percentage (20% by default) of volume from being available for configuring LUNs or for CIFS or NFS files to use. As Snapshot copies need space, they consume space in the snap reserve area. By default, after the snap reserve area is filled, the Snapshot copies start to take space from the general volume. Of course, because of WAFL technology, snap reserve does not actually reserve specific physical blocks for Snapshot usage and can be thought of as a logical space accounting mechanism. If the automatic Snapshot management in is not used, the recommended setting is zero.

Volume filling example

In this section, we present an example of how space is used from volume creation through filling up the volume with Snapshot data. In this example *lun reserve*, *fractional_reserve*, and *snap reserve* all have been left at their default values. (Figure 6 through Figure 9 illustrate this example.)

The steps of this example include:

1. A 100 GB volume is created. The available space is shown as 80 GB because by default there is a 20% snap reserve area.
2. A 30 GB LUN called *TestLUN* is created. Because LUNs are by default space-reserved objects, the 30 GB is immediately taken from the available space from the volume.
3. The LUN is filled with data and because the space was already reserved there is no change in reported space usage.

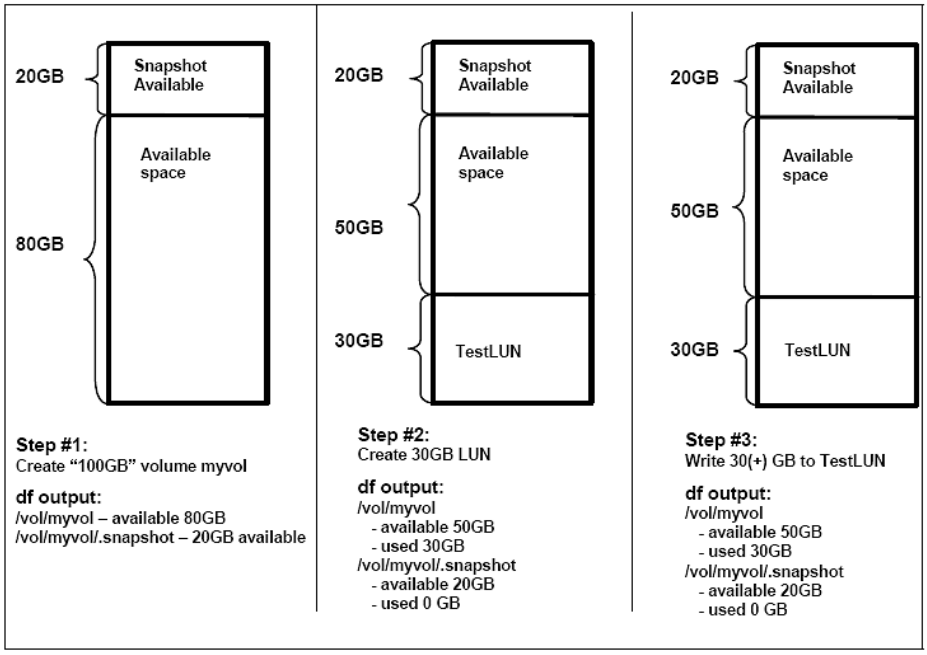

Figure 6 Volume filling: Actions #1 - #3

4. A Snapshot is created causing Data ONTAP to reserve 30 GB of space from the volume, ensuring that there is space for Snapshot data even if the LUN is completely overwritten. As shown in Figure 7, if `fractional_reserve` was set to less than 100, the amount of space reserved would be less than 30 GB.
5. 10 GB of data is overwritten in the TestLUN. Because a Snapshot exists, overwriting data in a LUN means that space is consumed for Snapshot data. The first space used for this is from the Snapshot reserve area. This is seen in the **`df`** output in the /vol/myvol/.Snapshot line where 10 GB is now shown as used.
6. Another 10 GB of data is overwritten in the TestLUN. This uses up the rest of the Snapshot reserve area, which now shows up as full.

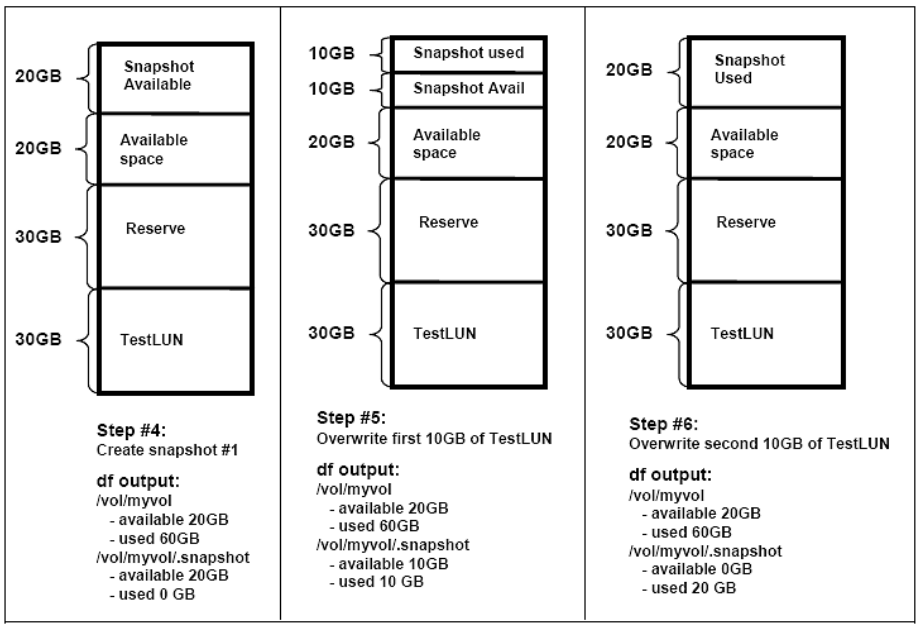

Figure 7 Volume filling: Actions #4 - #6

7. Another 10 GB of data is overwritten in the TestLUN. Even though the Snapshot reserve area is already full, the writes are allowed to happen and no Snapshot copies are lost or corrupted because there is still available space in the volume. These writes simply take space from the available space in the volume, which goes down from 20 GB as shown in the last step to 10 GB after Step #7. The Snapshot space usage that goes above the amount of snap reserve can be considered counted twice in the df command output because it shows up in the used column of both the general volume and the snap reserve space. It is interesting to note that the used column df output for the snap reserve area can go above 100%.
8. Another 10 GB of data is written to the TestLUN. In this case, because in the previous steps the entire LUN was already overwritten, no additional Snapshot data requires space. There is no change to the space usage.
9. A second Snapshot is created. Unlike the creation of the first Snapshot, the creation of more Snapshot copies does not cause any immediate changes to the space usage. The creation of additional Snapshot copies will not directly cause space to be consumed. However, as the count of Snapshot copies increases, the possibility that the Snapshot reserve space will be needed goes up, because multiple overwrites of a block are more likely to be caught in separate Snapshot copies.

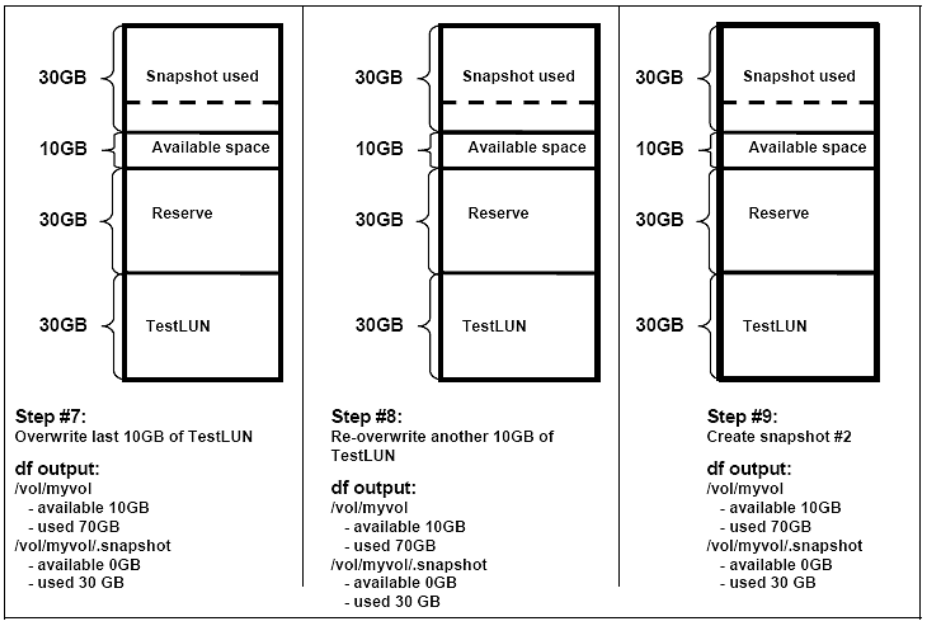

Figure 8 Volume filling: actions #7 - #9

10. 10 GB of the LUN is overwritten. Because a Snapshot copy was just created in the previous step, it does not matter which 10 GB of the LUN is overwritten, because space is still required to hold the Snapshot data. In this case, the 10 GB of available space is taken out of the volume. At this point the volume reports as full and it is not possible to create any more LUNs, create any files via NFS/CIFS, create more Snapshot copies, or write to non-space-reserved files.
11. An attempt is made to create a third Snapshot. This operation fails because the volume is full.
12. Another 10 GB of data is overwritten in the TestLUN. Even though the volume is full, the writes to the TestLUN are successful and the Snapshot copies are not lost because there is still space available in the reserve area. The only change to the df command output will be in the used column of the snap reserve area. Because the `fractional_reserve` is set to 100%, writes to the LUN will never fail for lack of space. If the `fractional_reserve` were

set to less than 100% and overwrites continued, at some point writes to the LUN might not be possible because of lack of space.

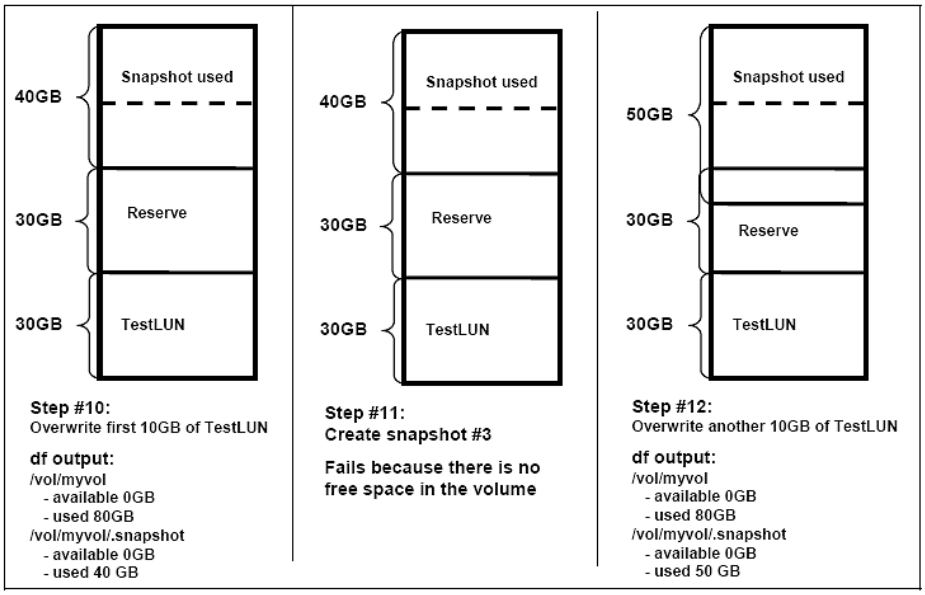

Figure 9 Volume filling: actions #10 - #12

This example shows how the reserve area is used only when there is no other space left in the volume.

Autosize

The *autosize* setting is a volume setting available in Data ONTAP that defines whether a volume should grow automatically to avoid filling up to capacity. This option is available only for flexible volumes. It is possible to define how fast the volume should grow with the **-i** option. The default growth increment is 5% of the volume size at creation. It is also possible to define how large the volume is allowed to grow to with the **-m** option. If volume autosize is enabled, the default maximum size to grow to is 120% of the original volume size (see Example 4).

Example 4 The autosize syntax

```
vol autosize <vol-name>  [-m <size>[k|m|g|t]]
                                                [-i <size>[k|m|g|t]]
                                                [ on | off | reset ]

     -m option sets the maximum size a volume is permitted to grow. Default
setting is 120% of volume size at the time of creation
     -i option defines the increment by which a volume is grown every time the
volume is full. Default setting is 5% of volume size at the time of creation
     vol autosize on – enables the feature.
     vol autosize off – disables the feature
     vol autosize reset – resets the settings to default value
```

Autodelete

The *autodelete* setting is a volume setting available in Data ONTAP that allows Data ONTAP to delete Snapshot copies if a definable threshold is met. This threshold is called a *trigger* and can be set so that Snapshot copies will be deleted automatically under one of the following conditions:

- *Volume*: The volume is near full. The volume reports that it is full even though there might still be space in the `snap_reserve` and `space_reserve` areas. This space is reported in the first line reported for each volume in the **`df`** command.
- *Snap_reserve*: The snap reserve space is near full. The snap reserve space reports being full and is considered independent of whether there is space in the `snap_reserve` and `space_reserve` areas.
- *Space_reserve*: The overwrite reserved space is full. This is the space determined by the LUNs with space reservations enabled and the `fractional_reserve` option. The reserve space will never be full until after both the volume and the `snap_reserve` areas are full.

There are three options (see Example 5) that the administrator can set to define the order in which Snapshot copies should be deleted:

- *Delete_order*: To determine whether the oldest or newest Snapshot copies should be deleted first.
- *Defer_deleted*: To allow the user to define a group of Snapshot copies that should be deleted first when no other Snapshot copies are available. It is possible to defer the deletion of user created or scheduled Snapshot copies or Snapshot copies beginning with a configurable prefix.
- *Commitment*: To determine how Snapshot copies used for SnapMirror® and dump operations should be handled. If set to `try`, it deletes only these Snapshot copies if they are not locked. If set to `disrupt`, these Snapshot copies are deleted even if they are locked. These will always be the last Snapshot copies deleted. In many configurations, deleting the last SnapMirror Snapshot copy is not desired because a new full baseline copy will be required to resume mirroring operations. If, for example, the source and destination are at different sites, this can be a time-consuming and costly process.

Example 5 The autodelete syntax

```
itsotuc> snap autodelete
     snap autodelete <vol-name> [on | off | show | reset | help] |
     snap autodelete <vol-name> <option> <value>...

     * Supported options and corresponding values are defined as follows:
        commitment                try, disrupt
        trigger                        volume, snap_reserve, space_reserve
        target_free_space       1-100
        delete_order                oldest_first, newest_first
        defer_delete                scheduled, user_created, prefix, none
        prefix                         <string>
```

The algorithm to determine which Snapshot copy to delete first looks for a Snapshot that does not lie in the `defer_delete` criteria and use the `delete_order` to determine whether to delete the oldest or more recent Snapshot. If no such Snapshot copy is found, the `defer_delete` criteria will be ignored in the selection process. If a Snapshot copy is still not available for deletion, then the SnapMirror and dump Snapshot copies will be targeted depending on the **`commit`** option.

Snapshot copies are no longer deleted when the free space in the trigger criteria reaches the value of the `target_free_space` variable, which defaults to 80%.

If autosize is enabled and the autodelete trigger is set to `volume`, the `try_first` volume option determines whether a volume grow or Snapshot copy delete is attempted first.

The two different types of thin provisioning

Given the widespread use of the industry-leading System Storage N series with Snapshot implementation, it is generally believed that there are two different types of thin provisioning:

- Thin provisioning of LUN space: The classic case of provisioning more space than is available.
- Thin provisioning of Snapshot space: If Snapshot copies are being used, having available less actual disk space than the total size of the LUNs will rarely, if ever, be practical. In this case the reserved space for Snapshot data is thinly provisioned.

Monitoring of available space

If thin provisioning is employed, it is imperative that the administrator monitor the free space on the storage system. If the available space for a critical application is getting too low, the administrator can execute a plan quickly and make more space available for that particular application. A well-managed system (that is, software is up to date and so forth) is not the same thing as a well-monitored system.

Thin provisioning of LUNs

Given the previous explanation of how LUNs will typically fill up close to 100%, strict thin provisioning of LUNs often will not provide significant savings for an extended period of time. However, with a large number of LUNs or very large LUNs, there is certainly the potential for savings.

There are many ways to configure the System Storage N series for LUN thin provisioning. Each configuration has its advantages and disadvantages. The following best practice configurations are recommended:

- Volume Guarantee = None Configuration (as shown in Example 6)

  `volume guarantee = none`
  `LUN reservation = not applicable`
  `volume auto_grow = off`
  `volume fractional_reserve` = Not applicable because Snapshot copies are not being used

  This configuration has the advantages that the free space in the aggregate is used as a shared pool of free space and that it works with all versions of SnapDrive® without issues.

  The disadvantage of this configuration is that the level of thin provisioning cannot be tuned on an individual volume basis.

  Example 6 Volume guarantee

```
vol options <vol-name> guarantee {none | file | volume}
  - set storage guarantee for volume <vol-name>
```

- Autogrow Configuration volume guarantee

  `Autogrow Configuration volume guarantee = volume`
  `LUN reservation = disabled`
  `volume auto_grow = on`
  `volume fractional_reserve` = Not applicable because Snapshot copies are not being used

  This configuration has the advantage that it is possible, if desired, to finely tune the level of thin provisioning for each application. With this configuration the volume size defines or guarantees an amount of space that is only available to LUNs within that volume. The aggregate provides a shared storage pool of available space for all the volumes contained within it. If the LUNs require more space than available in the volume, the volumes will automatically grow, taking more space from the containing aggregate. The degree of thin provisioning is done on a per-volume level, allowing an administrator to, for example, set the volume size to 95% of the cumulative LUN size for a more critical application and to 80% for a less critical application. It is possible to tune how much of the shared available space in the aggregate a particular application can consume by setting the maximum size to which the volume is allowed to grow.

  The disadvantage is that this configuration does not work with SnapDrive for Windows® (as of version 5.0 and earlier). Also, the current Data ONTAP limitation of 16TB aggregates should be considered in this scenario. The sum of all volumes on the aggregate should be sized in an autogrow configuration so that this limit is not approached.

- LUN Reserve Disabled Configuration volume guarantee (as shown in Example 7)

  `LUN Reserve Disabled Configuration volume guarantee = volume`
  `LUN reservation = disabled`
  `volume auto_grow = off`
  `volume fractional_reserve` = Not applicable because Snapshot copies are not being used

  This configuration is useful for traditional volumes and cases with multiple LUNs or LUNs for multiple applications in a single volume. For data management purposes in normal circumstances, having each application in separate volumes is recommended. However, since Snapshot copies are not used in this configuration, other considerations might lead to multiple applications in a single volume.

  This configuration ensures that the LUNs in a particular volume have access to the space reserved by the volume but limits those LUNs from consuming more than that amount of space. If more space is needed, the volume can still be grown manually using either the **`vol add`** command for traditional volumes or the **`vol grow`** command for flexible volumes.

Example 7 LUN create example

```
lun create -s size -t type [ -o noreserve ] lun_path
```

Thin provisioning through LUN growing

Because of the way that LUNs are used, one of the best ways to thin provision LUNs in a System Storage N series environment is to size the LUNs to what is needed at the time and then grow them as needed. Unlike other storage arrays, Data ONTAP enables this type of strategy with online, flexible, and simple provisioning from the storage side. In addition, the System Storage N series offers SnapDrive for Windows and SnapDrive for UNIX®, which completely automate the growing of the file system, taking care of all necessary actions on the host as well as on the storage. These technologies allow the storage administrator to grow the storage as needed while eliminating the over provisioning necessitated by solutions that do not enable quick and seamless LUN growth.

Thin provisioning of Snapshot space

System Storage N series customers take advantage of the industry-leading Data ONTAP Snapshot technology. Any time Snapshot copies are used, space needs to be reserved to hold the Snapshot data. The amount of space needed for Snapshot data varies greatly and is determined by a number of factors. A few of the most critical factors include application behavior, number of Snapshot copies being held, and the length of time for which they are held. For example, if Snapshot copies are being created solely by SnapMirror, the life of the copies is relatively short and a small number of copies are retained so that it is not necessary to have the default 100% reserve for the Snapshot space.

Thin provisioning of Snapshot space configurations

Given the large number of tunable options, there are a large number of possible configurations for thin provisioning of Snapshot space. Each configuration has various advantages and disadvantages. Here are recommendations for best practice configurations:

- **Autodelete Configuration**

```
guarantee           = volume
LUN reservation     = enabled
fractional_reserve  = 0%
snap_reserve        = Y%
autodelete          = snap_reserve / oldest_first
autosize            = off
try_first           = snap_delete
```

In this configuration, Snapshot copies are limited to using no more space than what's defined by `snap_reserve`. This is by default set to 20%. When the `snap_reserve` area fills up, the oldest Snapshot copy will be deleted to maintain free space. This configuration makes it a priority to keep the LUNs accessible over maintaining Snapshot copies.

One of the advantages of this configuration is that it is easy to monitor and understand the space usage by just monitoring the volume and reserve areas, seen under the **`df`** command as `.Snapshot`. The volumes are also independent of each other, meaning each application can be tuned independently. This configuration also has the advantage that only the oldest Snapshot copies are deleted while the most recent ones are maintained. The disadvantage of this configuration is that it does not use the available space in the aggregate as a shared pool of available space.

The `auto_grow` option could also be enabled for extra safety if, for example, SnapMirror is being used and those Snapshot copies should not be deleted. If those Snapshot copies cause the space to run out, the autogrow operation would allow the volume to take extra space from the aggregate to avoid running out of space.

- **Autosize Configuration**

```
guarantee           = volume
LUN reservation     = enabled
fractional_reserve  = 0%
snap_reserve        = 0%
autodelete          = off
autosize            = on
try_first           = volume_grow
```

The big advantage of this configuration is that it takes advantage of using the free space in the aggregate as a shared pool of available space. Because a guarantee of volume is being used, a different level of thin provisioning per application can easily be achieved through individually sizing the volumes and tuning how much each volume is allowed to

grow. Space usage is also very easy to monitor and understand by simply looking at the volume and aggregate usage.

As with all configurations using a shared free space, a disadvantage of this configuration is that the volumes are not 100% independent of one another since they are competing for that space when they need to grow.

- **Fractional_reserve Configuration**

```
guarantee           = volume
LUN reservation     = enabled
fractional_reserve  = Y%
snap_reserve        = 0%
autodelete          = space reserve / oldest_first
autosize            = on
try_first           = snap_delete
```

This configuration is similar to the default setting with a 100% setting for `fractional_reserve`, but it reduces the reserve space to below 100% and adds Snapshot autodelete as a final safety factor to prevent running out of space. The big advantage of this configuration is that SnapManager® for Exchange monitors the space available in the reserve area and delete SnapManager for Exchange-created Snapshot copies to free space as needed. SnapDrive for Windows (version 4.1 or later) can also monitor the space available in the space or fractional reserve area and issue alerts that allow the administrator to react.

Summary

When using System Storage N series with FlexVol technology, it is possible to thin provision both LUN and Snapshot space to allow greater efficiency and lower cost. The best practice configurations that we provide in this paper are excellent starting points. Administrators can modify these best practices as needed to fit a particular environment. There are multiple mechanisms in both Data ONTAP and higher level applications, such as SnapDrive and SnapManager for Exchange, that allow the administrator to customize the level of thin provisioning to match the environment, thereby enabling maximum cost savings.

The team that wrote this IBM Redpaper

This IBM Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), San Jose Center.

Alex Osuna is a Project Leader at the ITSO, Tucson Center. He writes extensively and teaches IBM classes worldwide on all areas of storage. Before joining the ITSO two years ago, Alex worked as a Principal Systems Engineer for Tivoli®. He holds over 10 certifications from IBM, Microsoft®, and Red Hat. He has over 29 years in the IT industry, 22 of them focused on storage.

Richard Jooss is a employee of the Network Appliance™ Corporation.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document REDP-4265-00 was created or updated on August 28, 2007.

Send us your comments in one of the following ways:
- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an e-mail to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ®	System Storage™
IBM®	Tivoli®

The following terms are trademarks of other companies:

Snapshot, FlexVol, Network Appliance, WAFL, SnapMirror, SnapManager, SnapDrive, Data ONTAP, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, Inc. in the U.S. and other countries.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.