EMC²
where information lives®

White Paper

EMC CLARiiON Reserved LUN Pool Configuration Considerations

Best Practices Planning

Abstract

This white paper provides guidelines and recommendations for global reserved LUN pool sizing and configuration, for use with SnapView™, SAN Copy™, and MirrorView™/Asynchronous on the CLARiiON® CX4, CX3, CX, and AX4 series storage systems.

September 2010

Part Number H1585.3

Table of Contents

Executive summary

Global reserved LUNs are used for SnapView™ sessions, SAN Copy™ incremental sessions, and MirrorView™/Asynchronous (MirrorView/A) mirrors. Given the storage-system-specific limits of the total number of reserved LUNs, proper sizing and layout will help to optimize the functionality of SnapView, SAN Copy, and MirrorView/A. General guidelines for configuration of reserved LUNs are based on expected change rates and duration of copy operations. Performance considerations also factor into proper configuration of reserved LUNs.

Reserved LUNs are required components of SnapView snapshot sessions, SAN Copy incremental sessions, and MirrorView/A mirrors. Each of these products maintains a consistent point-in-time data reference, and the role of the reserved LUN is to serve as a repository for this point-in-time reference.[1]

Reserved LUNs are assigned to source LUNs when the first session begins or an asynchronous mirror is created for that given source LUN. Additional reserved LUNs are assigned as needed; assignment is dynamic, and tracking of reserved LUN usage is available through Event Monitor. When all sessions for a given source LUN have been ended or the mirror destroyed, the associated reserved LUN(s) are free for assignment to another source LUN.

The storage administrator determines which LUNs are to be used as reserved LUNs by assigning LUNs to the reserved LUN pool[2]. A general guideline is that the reserved LUN typically needs to be 20 percent of the size of the source LUN. There are many factors to consider when determining the size of the reserved LUNs. Key among these factors, however, is the duration of a given session and the amount of source data that changes during a session.

Introduction

The following discussions provide guidelines to assist in determining best practices for reserved LUN sizing and configuration.

Audience

The intended audience for this white paper is members of the EMC field community who interact with customers and need to understand the reserved LUN considerations presented in this paper. Customers are also invited to read this paper to get a first-hand understanding of how to best size and configure their reserved LUNs.

Reserved LUNs and SnapView

SnapView snapshot sessions use reserved LUNs to support the pointer-and-copy-based design of SnapView. When the SnapView session starts, a *chunk map* keeps track of which chunks (groups of blocks) of data on the source LUN have been overwritten since the session started. As write requests are made to the source LUN, the data chunks are copied to the reserved LUN, which is a reserved area consisting of nonvolatile space on private LUNs. After this copy occurs, the write request to overwrite the source LUN data can be completed. Once each chunk is safely copied to the reserved LUN, the chunk map is updated with the new location of each of these chunks. This process—referred to as *copy-on-first-write*—occurs only when the first write request is made (after the session is started) to any chunk on the source LUN. Once the original data is safely copied to the reserved LUN, any number of writes can be made to the source LUN without losing the original contents (view) of the data.

The point-in-time view, therefore, is maintained by the pointers, which are directed to the original chunks as they exist either on the source LUN or the reserved LUN. To present this point-in-time view to a secondary server, a user may activate a *snapshot*, which may be thought of as a virtual LUN since the

[1] For more information on each of these products, refer to the following white papers: *CLARiiON SnapView Snapshots and Snap Sessions Knowledgebook - A Detailed Review, EMC CLARiiON SAN Copy - A Detailed Review*, and *MirrorView Knowledgebook*.

[2] In early revisions of SnapView, the reserved LUNs were called cache LUNs.

snapshot is reading the pointers to the data. Figure 1 shows the relationship among the source LUN, reserved LUN, and snapshot.

Figure 1. A SnapView snapshot as a composite of unchanged source data and data saved on the reserved LUN

In Figure 1, after the session starts, writes to the source LUN are designated with the prime character, such as *B'*, whose original chunk *B* has been copied to the reserved LUN.

SnapView session tracking

SnapView divides the source LUN into 64 KB chunks. This means that for a given server application write of 8 KB to the source, the corresponding 64 KB chunk will be copied into the reserved LUN—assuming this is the first update to this particular chunk of data after the session started.

As noted earlier, SnapView maintains a chunk map of the chunks that have been copied into the reserved LUN. The chunk map entry is copied to the reserved LUN as part of the copy-on-first-write procedure.

This chunk map is saved to disk for persistent sessions so that the data is preserved, even in the event of an SP trespass, reboot, or failure. Starting with release 24, all SnapView sessions are persistent by default. The space required for storing these chunk map entries is negligible; typically, it adds about 0.2 percent of the source LUN size. Thus, enabling sessions as persistent adds virtually no capacity requirement, and provides significant benefit for the user by remaining available even when the SP is unavailable or the system is rebooting.

Reserved LUN sizing considerations with SnapView

In determining the appropriate size of the reserved LUNs for SnapView sessions, users should consider the following.

SnapView source write activity

As shown in Figure 1, the reserved LUN(s) must be large enough to accommodate the copy-on-first-write activity on the source LUN. This means that the rate of change on the source LUN must be considered when sizing the reserved LUN(s) that are to be assigned to the reserved LUN pool. This rate of change will obviously vary by environment, so each user needs to make this determination according to the specific environment. A general rule of thumb is that a reserved LUN that is 10 to 20 percent of the size of the source LUN is probably sufficient for OLTP environments, where the read-to-write ratio tends to be around 2:1. In environments with a higher rate of change, larger reserved LUNs would be advised.

It is important to consider the pattern of the data writes. If the data has high locality of reference—that is, the same data areas tend to change many times during a session—then the copy-on-*first*-write design benefits users since subsequent writes to the source need not be rewritten to the reserved LUN. Additionally, if data is written sequentially, then the 64 KB tracking will correspond to multiple sequential writes, so that the amount of data that actually changes on the server maps more closely to the amount of data the storage system must copy. On the other hand, if the writes are very small (about 8 KB, for instance) and very random, then the 64 KB tracking will probably result in much more data being written to the reserved LUN than actually changed on the server. In environments with poor locality of reference and/or random writes, larger reserved LUNs should be used.

SnapView session duration

In addition to considering the write activity on the source volume during the SnapView session, it is also important to consider the duration of the session when sizing the reserved LUN(s). Because snap sessions can be created virtually instantaneously, some users create snap sessions on demand, using them for short, specific tasks (like backups); then, once the specific task is complete, the snap session is stopped. In other environments, however, users prefer to maintain snap sessions for longer periods of time—perhaps a day or longer. In this case, it's more likely that the copy-on-first-write operations will continue for a good portion of the session duration, and thus additional space will be required in the reserved LUN.

To determine how much data has been written during the session, users can view the session statistics (accessible from session properties), as shown in Figure 2.

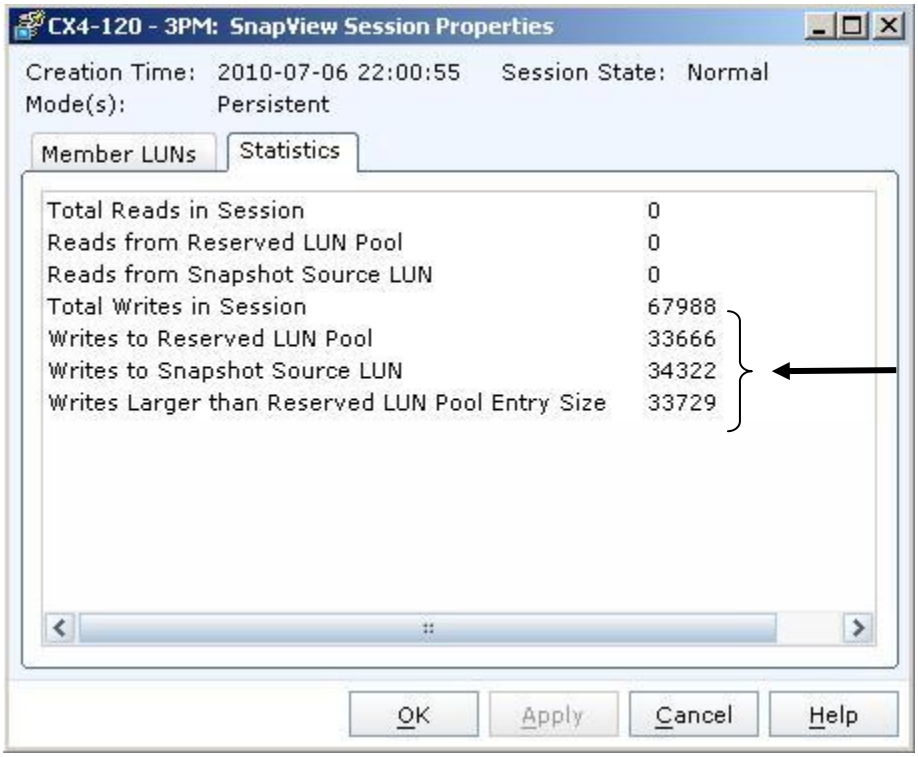

Figure 2. Copy-on-first-write data in a SnapView session

Navisphere® CLI also provides this information when you type the following command:

```
naviseccli -h <SP> snapview -listsessions -name <name> -totalwrites
```

As shown in Figure 2, the total writes that needed a copy-on-first-write can be viewed from the **Writes to Reserved LUN Pool** parameter, while **Writes to Snapshot Source LUN** lists the number of writes to the source LUN. These are writes to the source LUN that did not result in a copy-on-first-write operation. The sum of these two parameters is the **Total Writes in Session**.

The **Writes Larger than Reserved LUN Pool Entry Size** parameter refers to I/Os that are larger than the SnapView chunk size (which is 64 KB).

Snapshot write activity

The snapshot is both readable and writeable when presented to a secondary server. Any writes made to the snapshot are also stored in the reserved LUN, as shown in Figure 3. As with writes to the source LUN, snapshot write data chunk sizes are 64 KB.

Figure 3. SnapView snapshot writes are saved in the reserved LUN

Figure 3 illustrates that when the snapshot write is stored in the reserved LUN, it is stored in *addition to* (not *instead of*) the original data. In this example, B and B" are both stored in the reserved LUN; in other words the original data (B) has been modified on the source LUN and *also* modified on the snapshot (B''). When the snapshot is deactivated, the snapshot write (B") is deleted, but the original source data (B) is kept in the reserved LUN. This preserves the original point-in-time reference of the session, which allows the user to discard writes to the snapshot by simply deactivating and reactivating the same session. When the session is stopped, the original point-in-time data (B) is deleted and, if no other sessions are using the reserved LUN, the LUN is returned to the reserved LUN pool.

However, if the original data (C) has *not* been modified on the source LUN, but *is* modified to C'' on the snapshot, then the snapshot write (C'') is written to the reserved LUN pool. The original point-in-time reference of the session is still preserved simply by deactivating and reactivating the same session – which in this case would delete the snapshot write (C'').

Since snapshot writes are stored in the reserved LUN, along with the corresponding original data for that given chunk, anticipated snapshot write activity must be considered when sizing the reserved LUN(s). For example, given that the general guideline for the reserved LUN without snapshot writes is 10 to 20 percent, if every block of the snapshot is written, the reserved LUNs will need twice as much space to accommodate those writes. We recommend that users double the general guideline for each snapshot to which they plan to write. That is, a user with a single snapshot would probably calculate 10 to 20 percent of the source LUN for copy-on-first-write activity on the source, plus another 10 to 20 percent to accommodate the writes to the snapshot, resulting in a total of 20 to 40 percent the size of the source LUN.

Concurrent sessions

SnapView allows users to have up to eight sessions running on a given LUN. For instance, users may want to start a different session for every day of the week, as shown in Figure 4. Alternatively, users may want to start a different session every hour on the hour, then starting over again after the eighth hour, to have a rolling set of t-1 through t-8 sessions. With the *Rollback*[3] feature of SnapView sessions, it may also be very desirable to maintain multiple recovery points so that the user improves the chances of being able to recover to the desired point in time.

[3] SnapView Rollback allows users to recover data in the event of a software error—such as a data corruption—by copying the original data saved in the reserved LUN back to the source LUN. Refer to the *CLARiiON SnapView Snapshots and Snap Sessions Knowledgebook* white paper for details.

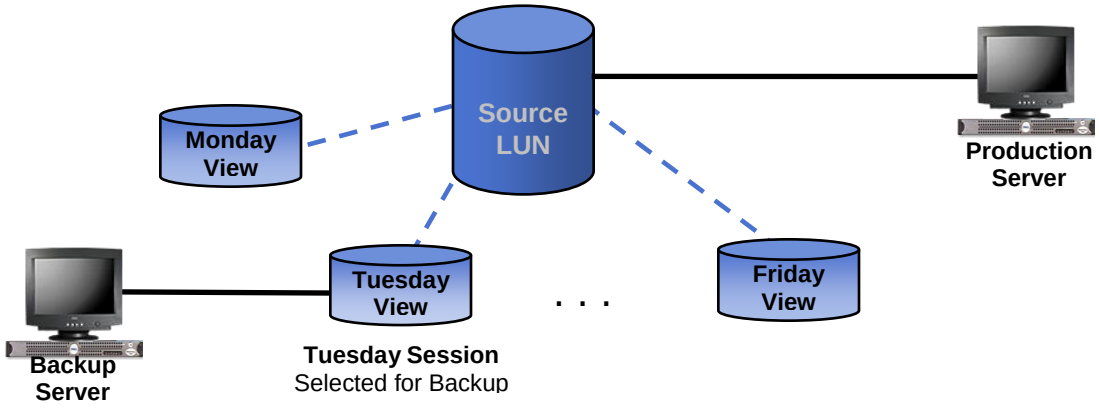

Figure 4. Concurrent SnapView sessions

Depending on the rate of change of the source LUN relative to when each session begins, additional reserved LUN space may be needed to accommodate multiple sessions. As Figure 4 shows, if there is very little change on the source LUN between the start of the Monday session and the start of the Tuesday session (and likewise through Friday), then the amount of additional reserved LUN space must be correspondingly small. On the other hand, if the source LUN changes most or all of its data between Monday and Tuesday (and likewise through Friday) then the amount of additional reserved LUN space will be correspondingly very large. Taken to an extreme, if users have eight sessions whose copy-on-first-write activity to the same set of chunks is completely different from one session to the next—that is, the same chunks must be rewritten with new data for each session—then it may be necessary to increase the earlier stated 10 to 20 percent guideline by a factor of eight. This would result in a guideline of 80 to 160 percent, where each of the eight sessions must maintain a completely different set of copy-on-first-write data.

Summary for SnapView usage of reserved LUNs

To summarize the guidelines just provided, users should consider the following when determining how to allocate reserved LUNs for SnapView sessions:

- Total size of the source LUN
- Rate of change of source LUN data
- Expected duration of SnapView sessions
- Expected snapshot duration and write rate
- Locality of source LUN and snapshot writes
- Expected concurrent session count
- Variation among size of source LUNs

Reserved LUNs and SAN Copy

SAN Copy uses SnapView components—sessions, snapshots, and reserved LUNs[4]—to support the incremental session feature made available in SAN Copy. The SnapView integration with SAN Copy provides a change-tracking mechanism, as well as a mechanism for maintaining a consistent point-in-time view for use during updates.

[4] If SnapView is enabled on the storage system, the SnapView components used by SAN Copy will be user-visible, but unavailable to the user. They will appear in the Navisphere UI as *reserved* objects: that is, *reserved snapshots* and *reserved sessions*, under the corresponding SnapView nodes in Navisphere. If SnapView is not enabled, these components will not appear.

SAN Copy and SnapView integration

When an incremental SAN Copy session is created, a SnapView session is started to keep track of the changes in the incremental session. As discussed earlier, a reserved LUN is assigned when the first SAN Copy session is created for a source LUN. Much like the SnapView chunk map, SAN Copy utilizes bitmaps to track changes between updates. These bitmaps are called the *tracking* and the *transfer* bitmaps. As the names suggest, one bitmap handles the tracking between updates, while the other bitmap directs the data transfers during the updates. As with the SnapView memory maps stored in the reserved LUN for persistent sessions, these delta maps are likewise stored in the reserved LUN, and are contained in the same structures that consume approximately 0.2 percent of the source LUN size.

The more significant use of reserved LUN space by SAN Copy incremental sessions is for the copy-on-first-write activity that occurs during a SAN Copy incremental update. During a SAN Copy incremental update, a snapshot is activated to the session, and the snapshot serves as the consistent source for data transfers to the destination LUN(s). This allows the SAN Copy source to stay online and accept incoming reads and writes for the duration of the SAN Copy incremental update. Incoming writes to chunks on the SAN Copy source LUN, which have been marked for transfer but have not yet been copied to the destination, generate a copy-on-first-write operation. As with SnapView sessions, this action stores the original chunk of data on the reserved LUN and redirects the associated pointer for that chunk so that the snapshot maintains a consistent view of the source LUN as it was when the update operation began[5].

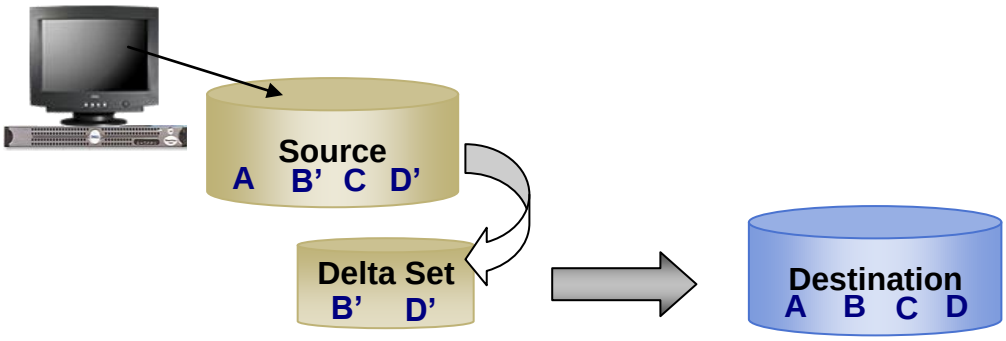

Figure 5. SAN Copy delta set for an incremental session

Once the update is completed, the copy-on-first-write activity ends, and the snapshot is deactivated and destroyed. The session continues, however, as it is tied to the delta maps.

Reserved LUN sizing considerations with SAN Copy

The initial reserved LUN that is assigned to the SAN Copy source LUN must be at least large enough to accommodate the delta maps in order to create the SAN Copy session in the first place. Once the session is successfully created and the update begins, additional reserved LUNs will be assigned as needed. If additional reserved LUNs are not available, the supporting SnapView session or point-in-time copy will be stopped, in turn causing the SAN Copy session to be unmarked. At that point, it will be necessary to allocate additional space in the reserved LUN pool, and then start a SAN Copy session that would create a new point-in-time copy and would mark the SAN Copy session; a full copy to the destination LUN(s) is not required. However, in order to avoid having under allocated reserved LUNs, the following considerations are provided to assist with proper sizing of reserved LUNs used by SAN Copy.

SAN Copy source write activity during initial synchronization

Please note that because the contents of the *entire* source LUN are copied to the destination LUN(s) during an initial synchronization (sync), it usually takes longer than an incremental update. Therefore, the most copy-on-first-write activity occurs during the initial sync. When you start a SAN Copy session, the session

[5] If the user issues an *explicit mark*, the copy-on-first-write activity begins right away. Only when the copy is subsequently started and then completed—or if the user issues an *unmark*—does the copy-on-first-write activity end.

then starts a SnapView session that runs for the duration of the SAN Copy session. The associated reserved LUN(s) assigned to the SAN Copy session will not be released until the SAN Copy session is destroyed. Because the most copy-on-first-write activity is likely to occur during the initial sync, any additional reserved LUNs required to support the copy-on-first-write activity during the initial sync will remain with the SAN Copy session until the session is destroyed.

To avoid this copy-on-first-write activity during the initial sync, you can use the **`-nomark`** option, which causes the initial copy to be read from the source LUN rather than from a consistent snapshot. As a result, the actual data transfer is much faster. The tradeoff, however, is that during the copy incoming I/O may cause the state of the data at the destination to be inconsistent. Consequently, it will be necessary to follow the **`-nomark`** initial copy with a marked incremental session to bring the destination to a consistent state.

The `-nomark` option is considered an advanced feature and is only available in Navisphere CLI.

You may want to schedule the initial sync during off-hours—for instance, overnight or over a weekend. That way, the copy occurs while there is little or no activity on the source LUN, and copy-on-first-write activity is minimized or eliminated during the update.

SAN Copy source write activity during incremental updates

When determining the appropriate reserved LUN size for SAN Copy incremental sessions, users should keep in mind that it is the copy-on-first-write activity during SAN Copy incremental updates that accounts for essentially all of the reserved LUN usage by SAN Copy (beyond the negligible space required for the delta maps). As with SnapView sessions, write activity on the source LUN—namely rate of change and randomness of writes—during the SAN Copy incremental update will drive the copy-on-first-write activity, and thus affect the required size of the reserved LUN holding the copy-on-first-write data. However, SAN Copy copy-on-first-write activity differs from SnapView sessions: The copy-on-first-write operation is only performed for chunks that have been marked for transfer but have not yet been copied when the write request is issued to the source LUN.

In addition to the write activity on the source LUN during the given update, it is also important to consider what related activities are occurring concurrently on this source LUN. The SnapView sessions that support SAN Copy incremental sessions count toward the maximum of eight SnapView sessions per source LUN. This means that users could have an additional seven SnapView sessions—or an additional seven SAN Copy incremental sessions or some combination, adding up to eight total sessions—concurrently. As discussed earlier, if the source LUN has multiple sessions with different copy-on-first-write activity, this will require additional reserved LUN space.

Duration of SAN Copy update to destination LUN(s)

The amount of time required to complete the update depends on the amount of data that must be transferred, as well as the selected transfer rate and available bandwidth. To determine how much data will need to be transferred, users can view the session status, as shown in Figure 6.

Figure 6. Data tracked for transfer for a SAN Copy incremental session

Navisphere CLI also provides this information when you type the following command:

```
naviseccli -h <SP> sancopy –info
```

If smaller data transfers are desired, users may opt to reduce the amount of time between updates, thus opting for shorter but more frequent transfers. On the other hand, if the source data has high locality of reference, then it may be more beneficial to increase the amount of time between updates, so as to avoid transferring the same dataset in each update. In either case, the user needs to understand the rate of change and locality of those changes on the source LUN, and how that will impact the amount of data being transferred. These factors will affect the amount of time that the copy-on-first-write activity must be supported.

Users can also impact the amount of time the transfer requires by increasing the transfer throttle speed or slowing down the transfer rate. Increasing the copy throttle consumes more storage processor CPU cycles to complete the transfer faster, whereas decreasing the copy throttle frees up storage processor CPU cycles for other tasks, thus allowing the transfer to occur more slowly over a longer period of time.

Finally, the actual bandwidth between sites will be a significant consideration regarding the transfer time. SAN Copy will use the bandwidth that is available; but, as with the throttle settings, users can direct SAN Copy to use only a portion of the bandwidth by reducing the bandwidth setting. Like the throttle adjustments, this setting allows the user to limit the data-transfer rate to conserve CPU resources. Limiting the data transfer rate increases the amount of time that data is transferred to the destination LUN(s), consequently increasing the amount of time that the copy-on-first-write activity is occurring.

For additional details on using and configuring SAN Copy, see the *EMC CLARiiON SAN Copy - A Detailed Review* white paper available on EMC.com and Powerlink (EMC's password-protected extranet for customers and partners).

Summary for SAN Copy incremental usage of reserved LUNs

When determining how to allocate reserved LUNs for SAN Copy incremental sessions, users should consider:

- SnapView session guidelines provided in the "Reserved LUNs and SnapView*"* section.
- Expected duration of SAN Copy updates to destination LUN(s), which will be based on the various factors and parameters discussed earlier.

The delta maps will only add approximately 0.2 percent the size of the source LUN to the space needed to accommodate copy-on-first-write during SAN Copy updates. Therefore, sizing reserved LUNs at 10 to 20 percent of the size of the source LUN (the general rule of thumb for SnapView sessions) would be appropriate for each SAN Copy session. If there are additional SnapView sessions, additional reserved LUN space will be needed.

Reserved LUNs and MirrorView/Asynchronous

As SAN Copy incremental sessions build on functionality provided by SnapView sessions, MirrorView/A mirrors build on functionality provided by SAN Copy and, therefore, SnapView[6]. As with SAN Copy incremental sessions, reserved LUNs are assigned to MirrorView/A primary image LUNs to provide tracking and copy-on-first-write during updates. In addition to this functionality, SnapView is further integrated with MirrorView/A to provide protection to the MirrorView/A secondary image LUNs during an update. This means that reserved LUNs must be allocated on both the primary and secondary storage systems to be able to create an async mirror.

MirrorView/A and SnapView integration

In addition to the SnapView integration discussed in the "Reserved LUNs and SnapView*"* section, MirrorView/A also uses SnapView to start a protective snap session on the MirrorView/A *secondary* (remote) images during each update (sometimes referred to as a *gold copy*). This protects the user in the

[6] If SnapView and SAN Copy are enabled on the storage system, the SnapView and SAN Copy components used by MirrorView/A will be user-visible but unavailable to the user. They will appear in the Navisphere UI as *reserved* objects—that is, *reserved snapshots* and *reserved sessions*—under the corresponding SnapView and SAN Copy nodes in Navisphere. If SnapView and SAN Copy are not enabled, these components will not appear.

event that the connection between storage systems becomes unavailable, and/or an error occurs on the MirrorView/A *primary* image LUN during an update. Without this protective snap session, a failure in the middle of an update could result in unusable data on the secondary storage system and perhaps also on the primary storage system.

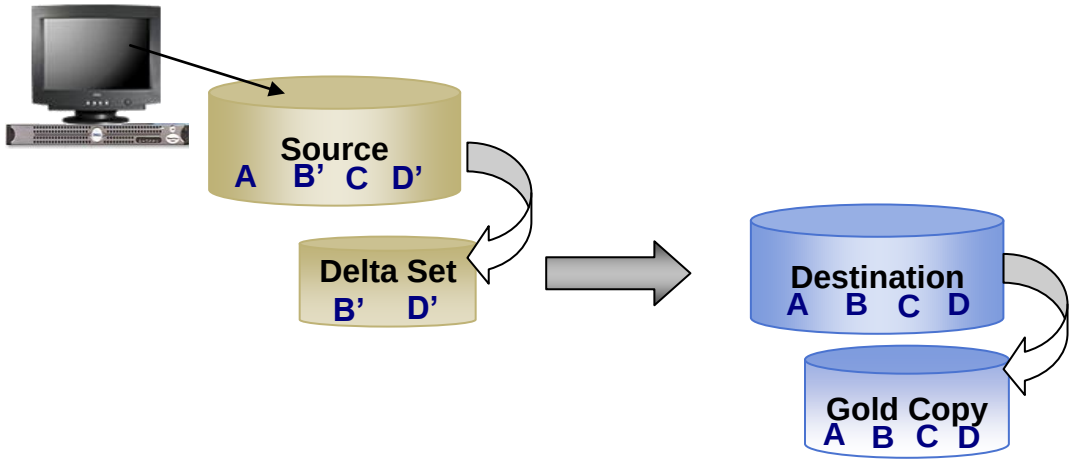

Figure 7. MirrorView/A delta set and protective snap session

If the update to the secondary image were stopped in the middle of the process, the data on the secondary image would be unusable since the update did not complete. Furthermore, if the primary image was no longer available for a subsequent update, then there would be no way to recover the data on either image. With the protective snap session, however, the user has the option of promoting the secondary image, at which point the snap session would be used to roll back the contents of the secondary image to the last successful update.

There is always a reserved LUN assigned to a MirrorView primary image. However, a reserved LUN is only assigned to the secondary image for the duration of the transfer. This is because a reserved snapshot session is explicitly started and stopped for each transfer.

When there is a MirrorView promote, these relationships are reversed; the secondary image becomes a primary image and a reserved LUN is assigned to the new primary image. The former primary (now secondary) image only has a reserved LUN assigned to it during updates.

Reserved LUN sizing considerations with MirrorView/A

The initial reserved LUN that is assigned to the MirrorView/A primary image LUN must be at least large enough to accommodate the delta maps in order to create the MirrorView/A mirror in the first place. Once the mirror is successfully created, the initial reserved LUN that is assigned to the MirrorView/A secondary LUN must likewise be at least 0.2 percent the size of the primary image LUN.

As with the SAN Copy update, once the MirrorView/A synchronization begins, additional reserved LUNs will be assigned as needed. Also like the SAN Copy update, if additional reserved LUNs are not available, the supporting SnapView session will be stopped, which in turn would cause the MirrorView/A mirror to be fractured—merging the tracking and transfer maps and freeing the reserved LUN pool data for that SnapView session. At that point, it will be necessary to allocate additional space in the reserved LUN pool, and start the MirrorView/A update. A full synchronization is not required since the merging of the tracking and transfer maps maintains the information on the difference in data between the primary and secondary images.

Meanwhile on the secondary storage system, additional reserved LUNs will be assigned to the secondary image as needed. If additional reserved LUNs are not available on the secondary storage system, the supporting SnapView session will be stopped, causing the mirror to fracture. At this point, the administrator will have to allocate additional reserved LUNs and resume the synchronization.

To avoid having under allocated reserved LUNs, the following information is provided to assist with proper sizing of reserved LUNs used by the primary and secondary images when using MirrorView/A.

MirrorView/A primary image write activity during copy

The reserved LUN space required for MirrorView/A primary LUNs follows the same guidelines as those for incremental SAN Copy sessions. The copy-on-first-write activity during the update accounts for the required reserved LUN space. The write activity (rate of change and randomness of writes) on the source LUN during updates will impact the copy-on-first-write operation. Additionally, any other SnapView or incremental SAN Copy sessions will require additional reserved LUN space for the given source LUN. In release 24 and later, the –nomark option is used by MirrorView/A for initial synchronizations. The –nomark minimizes copy-on-first-write activity during the initial synchronization.

Duration of MirrorView/A update to secondary image

As with SAN Copy, the anticipated duration of the MirrorView/A update will depend on the amount of data that must be transferred as well as available bandwidth. The amount of data to be copied can be found in the underlying SAN Copy session by using naviseccli. The following command will return the same "blocks to be copied" information as shown in Figure 6 on page 10.

```
naviseccli -h <SP> sancopy –info –reserved
```

The command can be used on any storage system that has both SAN Copy and MirrorView/A enablers installed, or any storage system running MirrorView/A that is at release 26 or later.

Similar to the throttle setting for SAN Copy, users can select the synchronization rate for their MirrorView/A mirrors. Synchronization rates are high, medium, and low; the default is medium. By increasing the synchronization rate, users speed up the data transfer, thus reducing the amount of time the copy-on-first-write activity must run on the primary image. Reducing the synchronization rate will slow down the data transfer rate, increasing the duration of the data transfer, and thus potentially increasing the copy-on-first-write. For update rate guidelines, see the *MirrorView Knowledgebook* white paper on EMC.com and Powerlink.

Protective snap session on the secondary image

Assuming no error on the primary image causes the update to fail, the user can size the reserved LUN according to how much data is expected to be copied to the secondary image during each update. A conservative estimate would double that size, in the event that one update fails (and assuming that the subsequent update succeeds). At the extreme, if users anticipate that more or less the entire LUN will be modified during the update, they may consider creating a reserved LUN that is the same size as the secondary image LUN since it will need to be large enough to accommodate copy-on-first-write operations for every (or nearly every) chunk of data on the LUN.

Summary for MirrorView/A usage of reserved LUNs

When determining how to allocate reserved LUNs for MirrorView/A mirrors, users should consider:

- SnapView session guidelines provided in the "Reserved LUNs and SnapView*"* section.
- SAN Copy session guidelines provided in the "Reserved LUNs and SAN Copy*"* section.
- Expected amount of data copied to the secondary image during updates.
- Expected duration of MirrorView/A updates to the secondary image.

To accommodate the 0.2 percent figure for delta maps—as well as the copy-on-first-write during the MirrorView/A updates—reserved LUNs at 10 to 20 percent the size of the source LUN (the general rule of thumb) would be appropriate for MirrorView/A. Additional space should be added for additional SnapView and/or SAN Copy incremental sessions running against those same LUNs.

As with SAN Copy, if the MirrorView/A primary LUN runs out of reserved LUN space, and there are not additional reserved LUNs available for assignment, the mirror must be destroyed and re-created with larger reserved LUN or LUNs allocated to the mirror. This requires a full resync when the mirror is destroyed and

re-created. If the secondary LUN runs out of space, the update is stopped immediately and resumes when additional reserved LUN space is available for assignment.

Performance considerations with reserved LUNs

In addition to considering the appropriate size for reserved LUNs, it is also important to consider the appropriate configuration of reserved LUNs. The following performance information is provided to help guide proper reserved LUN configuration.

Copy-on-first-write activity and effect on source I/O

The pointer-and-copy design of snapshots, while conserving disk space, typically affects source volume performance. This is partly due to the fact that when data is accessed that has not changed on the source volume, reads to the snapshot are accessing the same spindles as reads to the source volume. Furthermore, when a write request to the source initiates a copy of the original data to the reserved LUN, the CPU must perform the copy operation, thus decreasing the CPU processing cycles available to handle I/O to the source LUN.

Perhaps more significant is the load on the reserved LUN pool disks themselves. Copy-on-first-write (COFW) activity is comprised of two key operations: updating metadata in the chunk map and copying the protected data from the source to the reserved LUN. The update to the chunk map results in localized small-block writes to the map area on the reserved LUN, which is a very small percentage of the LUN capacity. The data chunk copy is a data read from the source LUN and subsequent write to the reserved LUN.

Overloaded reserved LUN pool disks result in higher latencies for COFW operations—which results in a higher response time for the host write that caused the COFW. So, some care must be taken to provision LUNs to the reserved LUN pool with sufficient *disks*, as well as capacity, since the load on the reserved LUN pool disks themselves is significant for performance.

For datasets with high locality of reference, the copy-on-first-write activity tends to subside fairly quickly. For example, in a database where various tables tend to receive the majority of the updates, once they have been updated the first time, there is no longer any copy-on-first-write to be performed for them. Figure 8 illustrates this effect. There is a significant spike in I/O response times (which correlates to the higher response times shown in the graph) for the initial copy-on-first-write operations, which affects performance on the source LUN. However, after some time has elapsed, the copy-on-first-write activity is significantly reduced, allowing the source LUN response times to more or less return to the baseline level shown at the far left of the graph (before starting the session).

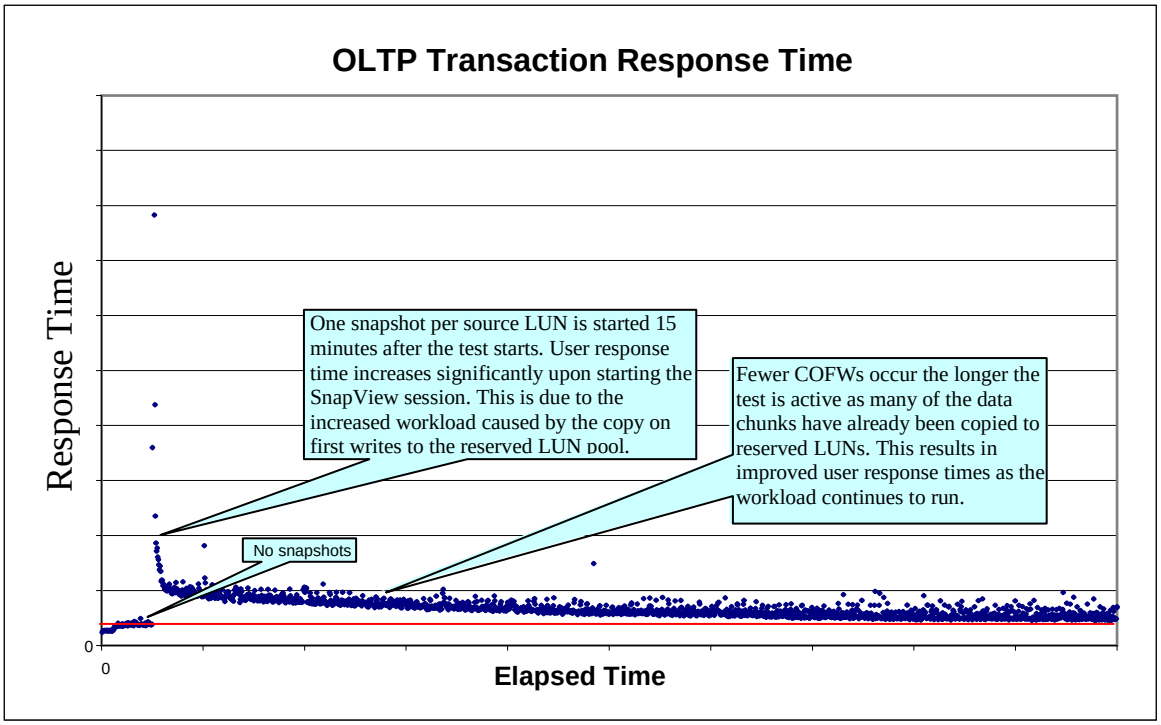

Figure 8. SnapView session recovery period

If users are using the snap session data for other purposes—such as backups—this creates additional spindle contention: The reserved LUN pool spindles are now working to satisfy the snapshot reads and writes, and are also performing copy-on-first-write operations. Also note that snapshots reads cause reads from the source LUN if the blocks on the source LUN have not been written to the reserved pool. Because of this, if at all possible, users are advised to allow the copy-on-first-write activity to subside before using the session data for secondary server I/O.

Flash drive considerations

Under certain conditions Flash drives can offer significant performance benefits over traditional, spinning drives. The most significant improvement occurs when the I/O profile has a component of small-block random reads. Applications like Microsoft Exchange and OLTP implementations of Oracle databases typically have these types of random read in their I/O profile.

As described previously, the COFW operation involves small-block metadata writes to reserved LUNs along with a chunk copy write. A high percentage of I/O activity on reserved LUNs is comprised of writes when COFW is being conducted. Therefore, Flash drives offer very little performance improvement over Fibre Channel for COFW operations when used for reserved LUNs. The best overall fit for reserved LUNs are Fibre Channel drives, no matter what drive type the source LUN may reside on.

Part of the COFW operation is a data chunk read from the source LUN. Flash drives can offer a significant benefit for this portion of the COFW operation, especially if the data chunk read results in a cache miss and the request must be serviced from disk. In cases where the chunk read is a cache miss, the read response time can be a significant portion of the overall COFW operation. So when the source LUN resides on Flash drives, the response time impact of the COFW operation can be reduced.

SATA drive considerations

Serial Advanced Technology-Attached (SATA) drives are economical because of their large capacities. However, they have slower rotational speeds and do not offer some of the advanced optimization features found in FC or SAS drives. They are designed for I/O profiles that have a duty cycle (the percentage of time that the drive is serving I/O) of less than 100 percent. They offer comparable performance to FC or

SAS drives when access profiles are sequential in nature. They are not the best choice for high levels of random access I/O.

The access patterns of reserved LUNs do not fit the ideal I/O profile for SATA drives. This is especially true if there are numerous LUNs on one RAID group. The access pattern becomes more random at the drive level when more reserved LUNs reside on a RAID group. This is due to increased concurrency.

SATA drives cannot absorb data from write cache as fast as other technologies when access patterns are random. Therefore, write cache is more apt to fill when reserved LUNs are placed on SATA drives. When this happens, performance is heavily impacted.

Reserved LUNs can reside on SATA drives for environments with modest COFW activity. This is most likely to be the case when the source LUNs also reside on SATA drives. We do not recommend that you place reserved LUNs on SATA drives when the source LUN resides on FC, SAS, or Flash drives. It may work for some environments; however, cache-full events are more likely to occur when reserved LUNs on SATA drives are asked to perform in lockstep with faster drive technologies.

LUN type, FAST, and FAST Cache

Traditional LUNs, RAID-group based LUNs, and thick LUNs are supported for reserved LUNs. Thick LUN support was added in release 30. Thin LUNs cannot be reserved LUNs. Thick LUNs are fully allocated in a pool at the time of LUN creation. COFW performance is slightly impacted by the mapping service for thick LUNs when compared to traditional LUNs. For the same source LUN, response time impact can be 10 percent higher when a thick reserved LUN is used as compared to a traditional reserved LUN.

LUN ownership is also important when using thick LUNs for reserved LUNs. It is generally recommended that you not change the default owner of a pool LUN. This is because the underlying structures used to provide storage to the LUN will still be controlled by the original SP. When SnapView takes a thick LUN from the reserved LUN pool, it tries to match LUN ownership between the source LUN and thick LUN. Therefore, you should ensure that there are an adequate number of thick reserved LUNs available for each SP. More information on pool-based (thick and thin) LUNs is provided in the *EMC CLARiiON Virtual Provisioning* white paper on Powerlink.

Release 30 adds the ability to configure pools with heterogeneous drive types. A single pool can have Flash, Fibre Channel, and SATA drive technology in any combination. EMC only recommends heterogeneous pools if you are using CLARiiON FAST for sub-LUN tiering. When thick LUNs are used as reserved LUNs in a heterogeneous pool, it is possible for the LUN to reside on any or all of the drive types in the pool.

We recommend that you use Fibre Channel and Flash drives for reserved LUNs whenever possible. When reserved-LUN data resides on SATA, it is possible that performance will be impacted more heavily. The default initial data placement in heterogeneous pools is auto-tier, which will initially spread the data out over all drive types. Data will be moved in subsequent relocation windows after its performance is measured relative to other data. Data relocation due to FAST will occur for reserved LUNs only if they are assigned to a session. The reserved LUN will not undergo any data movement if a reserved LUN is free (unassigned).

If you find that the default data placement and subsequent relocation are not satisfactory, you can select the highest tier and lowest tier for LUN data placement. You can use the highest tier to move LUNs to the highest drive tier in the pool; this may be helpful when SATA drives are present in a heterogeneous pool. You can use the lowest tier to move LUN data to the lowest tier in the pool; this may be helpful if a pool consists of Flash and Fibre Channel drives. When creating a new LUN, you can select the initial data placement under the **Advanced** tab in the **Create LUN** dialog box. After you create a LUN, you can see the current distribution of the LUN over each drive type. If you wish, you can change tiering preferences under the **Tiering** tab in the **LUN Properties** dialog box, as shown in Figure 9. More details about FAST can be found on the *EMC FAST for CLARiiON* white paper on Powerlink.

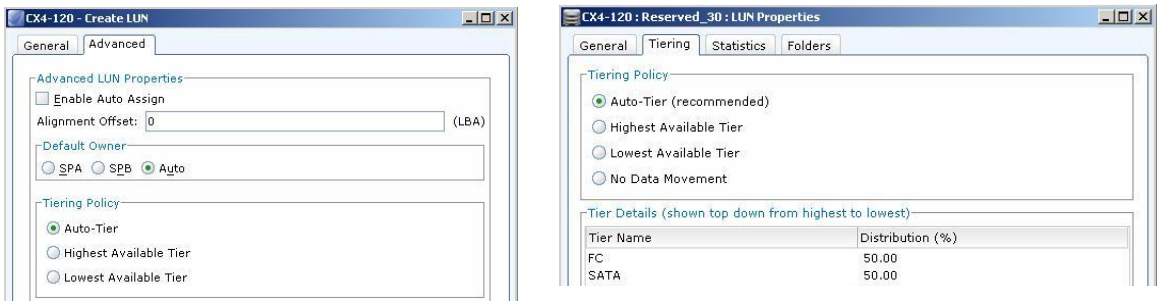

Figure 9. Create LUN dialog box, Advanced tab; LUN Properties dialog box, Tiering tab

FAST Cache is another optimization technology available in release 30. FAST Cache can, at a fine granularity, promote the most active data in the storage system to a cache of Flash drives. FAST Cache can be configured up to 2 TB in the CX4-960. You create FAST Cache at the storage-system level, and can enable or disable it at the traditional-LUN level and/or at the pool level. Once enabled at the system level, it will be enabled by default for any traditional LUNs or pools created subsequently. Since Flash drives do not provide a performance benefit over Fibre Channel drives for reserved LUNs, you can disable FAST Cache for traditional reserved LUNs. If a pool consists entirely of reserved LUNs, FAST Cache can also be disabled. However, if reserved LUNs reside in a pool with other server-visible LUNs, FAST Cache should remain enabled for the pool.

Using traditional LUNs, you can explicitly place the entire reserved LUN on any LUN type. Read cache and write cache are disabled by default on traditional LUNs that are created on Flash drives, while they are enabled by default on traditional LUNs created on all other drive types. If traditional LUNs on Flash drives are used for reserved LUNs, you should enable read and write cache. With read and write cache enabled, the performance of reserved LUNs on Flash drives and Fibre Channel drives is equal.

Recommendations for reserved LUNs

To minimize spindle contention during sessions, users should ensure that reserved LUNs are placed in the appropriate RAID group or pool for their usage.

Reserved LUNs on different physical drives than source LUNs

Reserved LUNs should *not* be created on drives serving source LUNs. This avoids the coordinated I/O patterns of host I/O and COFW operations. Additionally, when reserved LUNs are on different drives than their source LUNs, after COFW operations have occurred, snapshot reads and writes will have the least possible impact on the source LUN.

The effects of having reserved LUNs on the same spindles as the source LUN are most notable in RAID groups or equivalently sized pools ($\leq$ 16 drives). A single pool can contain all of the drives in the system except the vault drives and hot spares. In the cases of large pools, the workload is distributed over a larger set of drives, making it less likely that the slice being written to by the server is on the same drives as the active portion of its reserved LUN.

Reserved LUNs on the system drives

EMC recommends *not* putting response-time critical data on the first five drives of CX family storage systems because these drives contain the storage-system boot drives, FLARE® database, and vault drives. For the AX series, these elements exist on the first four drives. Heavy I/O to the LUNs configured on these drives could increase response time for Unisphere™ and Navisphere CLI. Also, for some systems any data LUN on these drives increases the time it takes to re-enable write cache after a vault drive failure. This is because after a replacement drive is inserted, any user data on the first five vault drives must be rebuilt before the write cache is enabled on the drives. CX4 series systems offer improved write cache availability and keep write cache active with a single vault drive failure. Hence, EMC recommends that the system drives not contain LUNs that are assigned to the global reserved LUN pool.

For detailed information on configuring the system drives, see the *EMC CLARiiON Best Practices for Performance and Availability* white paper on EMC.com and Powerlink.

Reserved LUNs properly balanced among RAID groups and pools

Having dedicated drives for global reserved LUN pool assignment allows ease of management and provides predictable I/O patterns on these dedicated drives. However, since access is restricted to the same group of drives, this can cause spindle contention on rotating drives and queuing due to copy-on-first-write or concurrent write activity for all SnapView sessions and snapshots. When reserved LUNs are configured by one of the replication wizards, the wizards will not configure more than 25 reserved LUNs per RAID group; for a pool it will not configure more than 25 reserved LUNs for each underlying RAID group in the pool. You should use the same guidelines when manually configuring reserved LUNs.

Preferably, users can distribute LUNs assigned to the reserved LUN pool across multiple RAID groups and pools to minimize spindle contention — generally, the more drives that are available to distribute the COFW activity, the better.

For example, certain drives that have user data can also accommodate the reserved LUNs. Preferably those drives can service the reserved LUN I/O activity and do not contain any LUNs that will be snapped. As a best practice, users may wish to configure no more than four to five reserved LUNs on such drives if they are contained in a RAID group or an equivalent-sized pool. As previously stated, pools distribute the load for each LUN across the entire pool, so the risk is diminished for larger pools.

Reserved LUNs with appropriate RAID type

Users should select a RAID type for their reserved LUNs that corresponds to the expected write activity on the reserved LUNs. RAID 5 tends to be a good general-purpose RAID type, and caters well to the copy-on-first-write activities reserved LUNs perform. If users elect to put their reserved LUNs on SATA drives, they should still use RAID 5, rather than RAID 3, which is selected at times for SATA. The copy-on-first-write operations do not perform well with RAID 3 since these drives will be randomly accessed in most situations.

Configuring the reserved LUN pool

All unallocated traditional LUNs and thick LUNs are available for selection to the reserved LUN pool. The reserved LUN pool is a "global" pool containing LUNs owned by SP A and SP B. When traditional LUNs are allocated by SnapView to a source LUN, their LUN ownership is changed[7] to match that of the source LUN. SnapView tries to select reserved thick LUNs that match SP ownership with the source LUN. If there are not any that match LUN ownership, one is assigned from the other SP. Assigning a thick LUN from another SP is supported, but this is not an optimal condition. You should be sure there are enough thick reserved LUNs for the number of LUNs on each SP.

Once LUNs are assigned to the reserved LUN pool, they will appear under the **Free LUNs** tab until they are assigned to a source LUN, at which point they will appear under the **Allocated LUNs** tab. Figure 10 and Figure 11 show the Reserved LUN tables for Free and Allocated LUNs, which can be seen by navigating to **Replicas** in Unisphere and clicking **Reserved LUN Pool**.

[7] Once allocated to the reserved LUN pool, the auto-assign flag is set for these LUNs (via LUN properties). This enables FLARE to trespass the reserved LUN when required to follow a source LUN trespass.

Figure 10. Reserved LUNs, Free LUNs tab

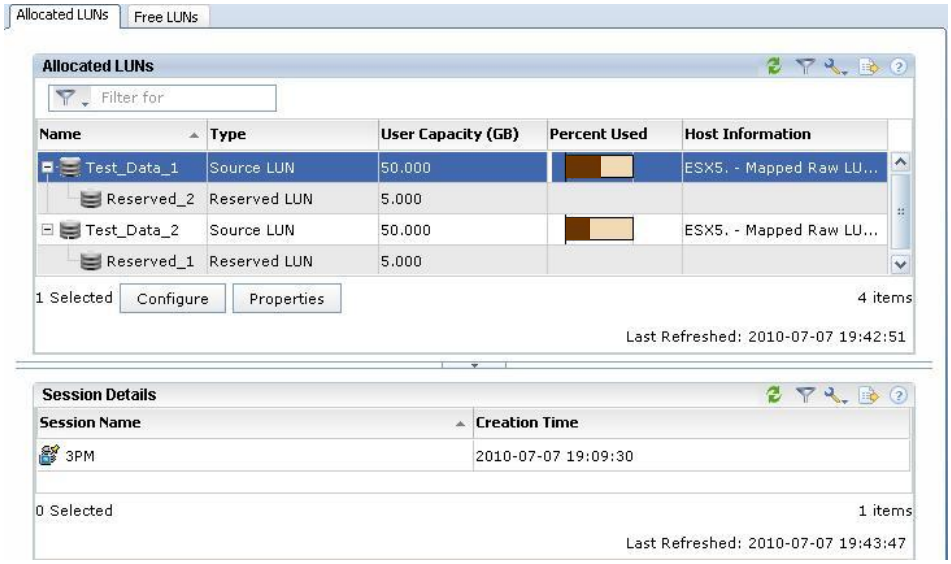

Figure 11. Reserved LUNs, Allocated LUNs tab

A series of wizards is also available within Unisphere to simplify the process of creating replicas and to assist in implementing best practices. The Snapshot wizard and Mirror wizard automatically create reserved LUNs when creating replicas. The SnapView wizard allows you to set the default reserved LUN size as a percentage of the source LUN capacity. The MirrorView wizard creates reserved LUN capacity equal to 20 percent of the capacity of the source LUN. Both wizards create two reserved LUNs for each source LUN equal to half of the reserved LUN capacity. For example, a mirror created with the wizard for a 100 GB LUN will create two 10 GB reserved LUNs.

By default, the wizards create thick reserved LUNs when possible. If this is not possible, the wizards create traditional LUNs. If possible, the reserved LUNs are placed in a storage pool or RAID group other than where the source LUN resides. The wizard also tries to avoid placing the reserved LUNs with other server-visible volumes. Lastly, the wizards consider how many other reserved LUNs reside in a storage pool or RAID group and will try to distribute the load as best as possible.

Similar information is available via Naviseccli. The first is snapview -lunpool:

```
naviseccli -h <SP> snapview –lunpool

Target LUNs      Associated LUN Pool LUNs    LUN Pool LUN Used Percent
13                       700
                         701                          80.934419

10                       11
                         12
                         14
                         702
                         7
                         8                          48.228936

9                        2                          2.797160
```

The snapview **–lunpool** output provides details about the number of reserved LUNs allocated to a given source LUN. In this example, SnapView has allotted two reserved LUNs (700 and 701) to source LUN 13.

A reserved LUN is assigned when you start the first session on a source LUN. The first available reserved LUN is assigned to that source LUN. If the reserved LUN becomes full while the session is running, the next available reserved LUN is automatically assigned to the source LUN. If there are no more reserved LUNs available in the reserved LUN pool, then the session that caused the reserved LUN to become full is stopped automatically. Any reserved LUNs being used by that session are returned to the reserved LUN pool, where they are available for use by other sessions.

The CLI command **`reserved –lunpool -list`** is also available to provide additional statistical information about the global reserved LUN pool. The following command produces the output shown below.

```
naviseccli -h <SP> reserved –lunpool -list
```

```
Name of the SP:                 SP A
Total Number of LUNs in Pool:       10
Number of Unallocated LUNs in Pool:  1
Unallocated LUNs:                15
Allocated LUNs:                  700, 701, 11, 12, 14, 702, 7, 8, 2
Total size in GB:               22.256470
Unallocated size in GB:          9.999756
Used LUN Pool in GB:             5.996704
% Used of LUN Pool:              26.943645
Chunk size in disk blocks:       128
```

The **reserved –lunpool list** command outlines the total space used by all LUNs within the global LUN pool, which is provided by the **Used LUN Pool** in GB parameter.

The **% Used of LUN Pool** parameter shows the percentage of space used by the reserved LUNs from the total space available for the entire global LUN pool.

To help users monitor the reserved LUN usage, SnapView sends an alert via Event Monitor when the reserved LUN(s) usage for a particular source LUN reaches 50 percent, then at 75 percent, and every 5 percent thereafter.

Users can also periodically check the percentage usage of the entire global LUN pool using the ***`reserved –lunpool –list`*** command.

If a user consumes the designated maximum number of reserved LUNs allowed per storage system, no additional reserved LUNs can be added when needed. As a result, sessions that need additional reserved

LUN space will not be able to obtain that additional space; therefore, those sessions will be stopped at the point at which they exceed the allocated reserved LUN space.

Reserved LUN sizing strategies

As noted earlier, a reserved LUN will be assigned to the source LUN as soon as the first session for that source LUN starts. If the reserved LUN runs out of space for storing the copy-on-first-write data, the next available reserved LUN in the reserved LUN pool will be dynamically assigned. The benefit of this automation is that manual intervention is not required any time a reserved LUN is to be assigned.

However, because all reserved LUN assignment is dynamic, users cannot control which reserved LUN(s) will be assigned to which source LUN. As such, the following suggestions may assist users in optimizing reserved LUN usage with SnapView sessions, SAN Copy incremental sessions, and MirrorView/A mirrors.

A common rule of thumb is to allocate reserved LUN capacity equal to 20 percent of source LUN capacity. This guideline is commonly used in the early stages of planning where the amount of copy-on-first-write activity hasn't been measured for a given workload. Note that for thin LUNs, you should not base your estimates on thin LUN user capacity, because there is usually some level of oversubscribed capacity. You should base their estimates either on the percentage of the storage pool that will be snapped or on the total size of the storage pool. The following sections describe various methods for allocating reserved LUN count and capacity based on varying source characteristics.

Uniform sizing of reserved LUNs

Since reserved LUNs are automatically assigned, users are encouraged to select a uniform size for reserved LUNs, so that all reserved LUNs are suitable for use with any source LUNs. This is especially helpful in environments where the use of SnapView, SAN Copy incremental sessions, or MirrorView/A sessions is fairly random and in flux, since the dynamic assignment of reserved LUNs will be in full force.

Note that the minimum size of the reserved LUN should be large enough to hold the bitmap of the LUN to be snapped. The minimum size of the bitmap is about 0.2 percent of the size of the source LUN. For thin LUNs, this is based on user capacity. Hence, take into account the bitmap space required by the reserved LUN when you configure the reserved LUNs.

For environments with disparate source LUN sizes, users must determine whether the uniform size will be based on the smallest or largest source LUNs. In the examples shown in Table 1, an environment of ten 100 GB LUNs and two 500 GB LUNs is used.

Small reserved LUNs to conserve space

If storage-system capacity is an issue, users should select a uniform size based on the size of the smallest source LUN. This means that the larger source LUNs will likely need multiple reserved LUNs. Since reserved LUN assignment is dynamic, SnapView will take care of the assignment of multiple reserved LUNs as needed.

Using the guideline that the reserved LUN usage runs at 20 percent the size of the source LUN, and applying that figure to the smallest source LUN, the reserved LUNs would be 20 GB. Table 1 estimates how the LUN assignments would go.

Table 1. Example of using small uniform reserved LUNs

Source LUN Size	Reserved LUN Size	Reserved LUN per Source	Total Source LUNs	Total Reserved LUNs
100 GB	20 GB	1	10	10
500 GB	20 GB	5	2	10
Reserved LUNs				**20 @ 20 GB**

In this example, the smaller source LUNs require a single reserved LUN (because the reserved LUNs are all 20 percent the size of the smaller source LUN size), whereas the larger source LUNs require more reserved LUNs because they are larger. The total reserved LUN usage is 20 reserved LUNs (for a total of 12 source LUNs), each at 20 GB. Thus, this solution implements a larger total number of reserved LUNs. But, because they are small reserved LUNs, the total space used is minimized: 20 x 20 GB = 400 GB.

Large reserved LUNs to maximize participating source LUNs

At the same time, users should keep in mind that there is a limit to the total number of reserved LUNs per storage system[8]. Since each source LUN requires at least one reserved LUN, these limits effectively translate to the number of source LUNs on which a session can be started. If users desire to start sessions on as many source LUNs as possible, then they will want to ensure that the larger source LUNs are not consuming multiple reserved LUNs. Therefore, in environments where users want to maximize the number of source LUNs that can have sessions running on them, the uniform size should be based on the *largest* source LUN size. This ensures that each source LUN requires only a single reserved LUN, and thus more source LUNs can run by running sessions concurrently.

Table 2. Example of using large uniform reserved LUNs

Source LUN Size	Reserved LUN Size	Reserved LUN per Source	Total Source LUNs	Total Reserved LUNs
100 GB	100 GB	1	10	10
500 GB	100 GB	1	2	2
Total Reserved LUNs				**12 @ 100 GB**

In this example, the reserved LUNs are large enough so that the larger source LUNs require only a single reserved LUN. But, the smaller source LUNs have reserved LUNs that are 100 percent the size of the source LUNs, which is likely going to be far more space than is needed. The total reserved LUN usage is 12 reserved LUNs, but the total space used is 12 x 100 GB—or 1200 GB. This solution minimizes the total number of reserved LUNs used but requires much more space than the previous solution.

These examples show two ends of the spectrum of uniform sizing of reserved LUNs. The first example required a higher total count of reserved LUNs, but minimized space consumed by reserved LUNs. The second example is the converse, where fewer total reserved LUNs were required, but three times the space was consumed. Users must decide whether they are constrained by the total number of reserved LUNs or the total amount of space that they have available for use.

Stepwise configuration

Although users cannot explicitly control which reserved LUNs get assigned to which source LUN, it is possible in some environments to achieve some control on the assignment process by stepwise configuration of resources. This essentially forces assignment as desired. These environments tend to be much less random, so that users can more or less dictate usage patterns and required space to support those patterns.

For instance, given that a reserved LUN is assigned when the first session is started for a source LUN, some users elect to use a procedure like the following example to control the reserved LUN assignment:

1. Create LUN 101 where size = 2 GB.
2. Assign LUN 101 to the reserved LUN pool.
3. Start session 1 on source LUN 1 where size = 10 GB (5x the reserved LUN).

[8] Users may have up to 512 reserved LUNs on a CX4-960, up to 256 reserved LUNs on a CX4-480, and up to 128 reserved LUNs on both a CX4-240 and CX4-120.

Reserved LUN 101 is assigned to source LUN 1 because it is the only available LUN in the reserved LUN pool.

4. Create LUN 105 where size = 200 GB.
5. Assign LUN 105 to the reserved LUN pool.
6. Start session 1 on source LUN 5 where size = 1 TB (5x the reserved LUN).

Reserved LUN 105 is assigned to source LUN 5 because it's the only available LUN in the reserved LUN pool.

This procedure would work especially well in a SAN Copy incremental or MirrorView/A environment where the reserved LUN will never be released to be used by any other session. SnapView sessions, on the other hand, tend to be more dynamic since they are released when a session is stopped if you are routinely starting/stopping sessions.

Users should keep in mind that, unless they feel sure that their environment caters to the predictability required for the stepwise configuration, the uniform sizing of reserved LUNs is much less involved and generally a simpler and more common implementation.

Conclusion

Reserved LUNs support the copy-on-first-write activities of SnapView sessions, SAN Copy incremental sessions, and MirrorView/A mirrors, providing a consistent point-in-time copy of data. Each product has generally the same guidelines for reserved LUN consumption: size of source LUN, rate of change of data on the source LUN, as well as expected duration of the session. At the same time, there are various specific considerations for each product that must factor into the correct sizing of reserved LUNs.

The dynamic assignment of reserved LUNs assists users by ensuring that, should reserved LUN space be consumed, the reserved LUNs will be assigned as needed (and as available), thereby ensuring that there are sufficient reserved LUNs in size and quantity as is necessary to support this feature.