

Designing and Optimizing Dell/EMC SAN Configurations **Part 2**

Dell/EMC storage area networks offer storage architects and administrators several design options and optimization settings. In the second of a two-part series, members of the Dell Server and Storage Performance team address the performance aspects of binding logical storage units, or *LUNs*, and provide an introduction to *metaLUNs*.

BY **ARRIAN MEHIS AND SCOTT STANFORD**

Dell/EMC storage area networks (SANs) are called upon to serve networked data for a wide variety of enterprise applications, and can benefit from optimized, well-planned storage processor (SP) and logical storage unit (LUN) designs. In the first installment of this two-part tutorial on Dell/EMC SAN design and optimization,¹ best practices for designing LUNs and optimizing SP cache settings were covered in depth. Part 2 concludes the series with additional guidance on binding LUNs and an introduction to the concept of *metaLUNs*.

Binding LUNs: RAID level

The most common RAID types used in production IT environments are RAID-1, RAID-5, and RAID-10. However, as explained in Part 1, fault-tolerant RAID types typically provide reduced performance as compared to non-fault-tolerant RAID types. Best practices suggest that administrators select a RAID type that suits the performance demands of the workload yet offers data integrity in case a drive fails. RAID-5 is popular because it provides a balance of performance, data integrity, and capacity. However, because RAID-5 involves parity, a write penalty is incurred. RAID-5 is well suited for:

- Data mining
- Medium-performance media serving
- Messaging applications such as e-mail

- Any application in which the write load does not cause cache saturation
- Database log activity

RAID-10 can be an excellent choice for performing random writes. The disk segments are mirrored, so the system can read from either segment but must write to both segments. RAID-10 is well suited for:

- Microsoft® Exchange and similar applications
- Real-time data and brokerage records
- Database tables containing small records that are updated frequently

RAID-0 can provide optimal performance because of its individual disk access and zero redundancy. It also offers the highest storage capacity of all the RAID levels. However, the trade-off for maximum speed and capacity is the risk of data loss. RAID-0 is well suited for:

- Temporary directories
- Any volume for which maximum speed is important and data loss does not matter

RAID-1 can provide a dedicated file system when capacity needs are small enough to make RAID-10 a

¹ For the first part of this two-part article series, see "Designing and Optimizing Dell/EMC SAN Configurations: Part 1" by Arrian Mehis and Scott Stanford in *Dell Power Solutions*, June 2004.

RAID level	Disks in array	Effective disks	Element size	Stripe size
RAID-5	6	5	64 KB	320 KB
RAID-0	10	10	16 KB	160 KB
RAID-10	8	4	32 KB	128 KB

Figure 1. RAID level, element size, and stripe size matrix for Dell/EMC CX series arrays

less-effective solution. For example, RAID-10 requires a minimum of four drives (actual capacity of $n/2$) whereas RAID-1 has a minimum and maximum of two drives (actual capacity of $n-1$). Because RAID-1 is a mirrored configuration, it is well suited for:

- Operating systems in which speed is not a factor
- Systems that require data integrity but have only two available drives
- Cluster quorum resource LUNs

Refer to *EMC CLARiiON Best Practices for Fibre Channel Devices* at https://powerlink.emc.com/HighFreq/folder_0/H519.3_clariion_fibre_chnl_dev_ldv.pdf for more information about RAID levels.

Binding LUNs: Element size

Element size is the size in disk blocks (sectors) of the area on each disk that a storage system can read or write to without accessing another disk (assuming that the transfer starts at the first sector in the stripe). Element size can be referred to in terms of elements or actual bytes. Because Dell/EMC storage arrays running EMC® Navisphere® software (release 12 or later) use 512 bytes per block, an element size of 2 represents a physical size of 1 KB, an element size of 4 represents a physical size of 2 KB, and so forth. The default element size for Navisphere is 128 (64 KB).

The physical translation is important when selecting LUN element size. The smaller the element size, the more efficient the distribution of data read or written. However, if the size is too small for a single I/O operation, the operation requires access to two stripes, which requires reading and/or writing from two disks instead of one. This is known as disk crossing. Best practices recommend selecting a size that is a multiple of 16 sectors (8 KB) and is the smallest size that will rarely result in forced access to another disk.

Be aware that most writing logic will write to the same stripe multiple times if more than one data size will fit. Writing multiple data sizes to one disk is much more efficient than writing one data size per disk. Note that this technique is not the same as disk crossing.

Element size also directly dictates the stripe size of an array. For example, a RAID-5 array comprising six disks (five effective disks plus one parity disk) with a 64 KB element size would have a stripe size of 320 KB (5×64 KB). Likewise, a RAID-0 array comprising

10 disks (10 effective disks) with a 16 KB element size would have a stripe size of 160 KB (10×16 KB). Furthermore, a RAID-10 array comprising eight disks (four effective disks) with a 32 KB element size would have a stripe size of 128 KB (4×32 KB). Figure 1 summarizes these example configurations.

Binding LUNs: Alignment offset

The host via its operating system may record private information, known as a signature block, at the start of a RAID-0, RAID-5, or RAID-10 stripe. The alignment offset can be used to force the beginning of usable LUN space to start past the signature block. The signature block size, which is dictated by the storage array manufacturer (EMC, in this case), consumes only 16 KB of space, but the NT file system (NTFS)—a file system for Microsoft Windows® operating systems—reserves 31.5 KB of space for the signature block. This reservation of extra space at the beginning of the LUN can increase the risk of disk crossings or stripe crossings. Stripe crossings are similar in principle to disk crossings, but refer to accessing two stripes instead of one. If administrators do not adjust the NTFS signature block size from its default of 31.5 KB to 16 KB, the stripe will not align with the signature block and data will be written to two stripes instead of one.

For example, because NTFS reserves 31.5 KB of signature space, if a LUN has an element size of 64 KB with the default alignment offset of 0 (both are default Navisphere settings), a 64 KB write to that LUN would result in a disk crossing even though it would seem to fit perfectly on the disk. A disk crossing can also be referred to as a split I/O because the read or write must be split into two or more segments. In this case, 32.5 KB would be written to the first disk and 31.5 KB would be written to the following disk, because the beginning of the stripe is offset by 31.5 KB of signature space. This problem can be avoided by providing the correct alignment offset.

Each alignment offset value represents one block. Therefore, EMC recommends setting the alignment offset value to 63, because 63 times 512 bytes is 31.5 KB. Contact Dell or EMC to determine the latest recommended alignment offset value, because improper use will degrade performance.² If the alignment offset requirement is uncertain, leave it at the Navisphere default.

Verifying priority

Once a LUN is bound, Navisphere will perform an *Initial Verify* action to search the entire LUN for latent soft media errors and eliminate them. The Initial Verify action can be disabled with the No Initial Verify checkbox when binding a LUN. It is a good practice to perform the Initial Verify step when binding a LUN. However, bear in mind that the verify rate is about 4 GB/minute, which can be time-consuming on a LUN that is several terabytes in size. It is preferable not to send any data to the LUN until the background verify operation is complete and I/O is idle.

²For current recommended offset values, see *EMC CLARiiON Best Practices for Fibre Channel Devices* at https://powerlink.emc.com/HighFreq/folder_0/H519.3_clariion_fibre_chnl_dev_ldv.pdf.

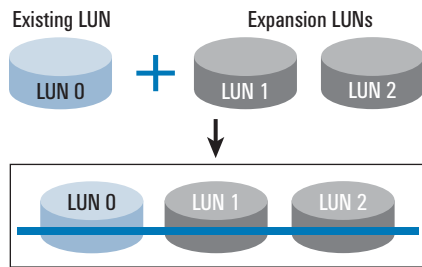


Figure 2. LUN stripe expansion

The *Verify Priority* action checks parity sectors in a LUN. If an SP detects parity inconsistencies, it will initiate a background process to check all parity sectors in the LUN. The Verify Priority action dictates the amount of resources that the SP devotes to checking parity instead of normal I/O activity. By default, the priority is set to ASAP (as soon as possible), but can be modified to High, Medium, or Low priority when binding a LUN or after LUN creation. ASAP and High priorities will rebuild faster, but consequently will affect storage system performance more than Medium and Low priorities. The Verify Priority option is not available for RAID-0, disk, or hot-spare binds.

The *Rebuild Priority* action is much like Verify Priority except that it specifies the rate at which to reconstruct data on a hot spare or new disk module that replaces a failed disk module in a LUN. Again, this option is not available for RAID-0, disk, or hot-spare binds.

The other aspect of these verify actions is the continuous checking, or *sniffing*, for soft media errors throughout the lifetime of the LUN. This is indicative of the polling disk activity that occurs at a constant interval on the LUN. The default sniffer rate is 5 milliseconds per I/O. This rate can be changed using the Navisphere command-line interface (CLI). Although the polling disk activity may appear high, it is normal and preferable to leave it at the default setting because there is no actual I/O on the LUN that would interfere with performance. This option is available for binds of all RAID levels.

Setting cache options

As described in Part 1 of this article series, SP cache settings should be configured according to the type of workload. Read cache should be enabled or disabled based on specific workloads. Correlating with read cache being enabled or disabled, cache prefetch should be enabled or disabled based on specific workloads. The type of workload also should determine whether write cache is enabled or disabled; however, most workloads usually will benefit from the latency reduction advantages provided by write cache.

Understanding metaLUNs

Up to this point, the concepts of mapping LUNs to RAID groups and LUNs to SP symmetry has focused on a one-to-one relationship. MetaLUNs introduce another layer of control above FLARE™ LUNs, which are normal LUNs created in the EMC FLARE operating

environment. MetaLUNs act much like software RAID on the operating system, except the storage array handles the overhead instead of the server. MetaLUNs offer a method for expanding—through striping or concatenation—the FLARE LUNs beyond their 16-spindle capacity into a single, manageable metaLUN. Striping or spanning of LUNs is extremely beneficial when large capacity is needed or performance requirements demand a high spindle count.

Best practices for using metaLUNs

A LUN or metaLUN can be expanded in two ways: *stripe* expansion or *concatenate* expansion. A stripe expansion takes the existing data on the LUN or metaLUN that is being expanded and re-stripes (redistributes) it across the existing LUNs and the new LUNs being added. This stripe expansion may take a long time to complete. With stripe expansion, only FLARE LUNs of the exact same size and type may be bound as a metaLUN. Figure 2 depicts how stripe expansion works.

A concatenate expansion creates a new metaLUN component that includes the new expansion LUNs and appends this component to the existing LUN or metaLUN as a single, separate striped component. No re-striping of data between the original storage and the new LUNs occurs. The concatenate operation completes immediately. With concatenate expansion, FLARE LUNs of differing size and type may be bound as a metaLUN. Figure 3 depicts how concatenate expansion works.

A metaLUN also has an *element size multiplier* feature. By default, the element size multiplier is set to 40. This may be changed to any value from 1 to 255. For example, using the RAID-5 array listed in Figure 1, which is composed of six disks with a 64 KB element size (320 KB stripe size), an element size multiplier of 40 would result in a metaLUN stripe size of 12.8 MB (40 × 320 KB).

Following the LUN symmetry practices described in Part 1 of this article series, when expanding LUNs into a metaLUN, it is preferable to select LUNs that reside on alternating back-end loops (except on CX200 and CX300 storage arrays). Doing so will allow the storage processor that owns the metaLUN to duplex the reads and writes across the back-end loops. This ultimately provides better load balancing for the storage system and avoids the potential for a bus bottleneck.

For example, using the symmetrical LUN design for the hypothetical small business described in Part 1—in which a CX400 array was configured with six LUNs for three small workloads, two

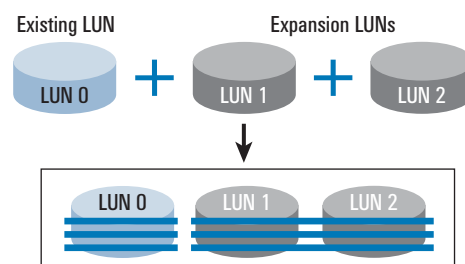


Figure 3. LUN concatenate expansion

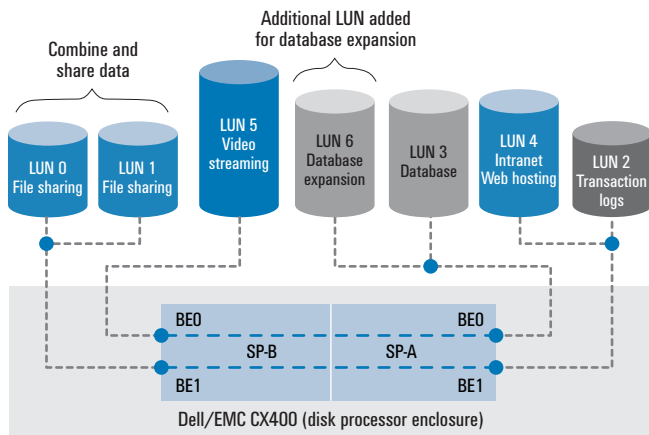


Figure 4. Expansion LUN staging for hypothetical small-business scenario

medium workloads, and one large workload—imagine the following scenarios:

1. The company’s accounting department, which owns LUN 0, has just merged with the finance department, which owns LUN 1. For security, data integrity, and access control reasons, they both want to share data in a single file-sharing LUN instead of two separate volumes. The administrator must combine the LUNs, which are two different capacities, while preserving the existing data.
2. The sales department, which owns LUN 3, realizes that its database is reaching maximum capacity and therefore must be increased in size while maintaining performance. So the administrator creates a LUN of the exact same size and type (LUN 6) for expansion.

Figure 4 shows the expansion LUN staging for this storage array. Using the metaLUN feature in Navisphere, the administrator can satisfy the wishes of the accounting and finance departments by performing a concatenate expansion of the two FLARE LUNs of different sizes. Also, the administrator can address the demands of the sales

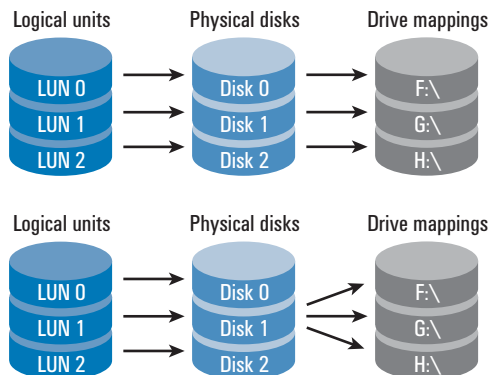


Figure 6. Partitioning LUNs as one mapping or multiple-drive mappings

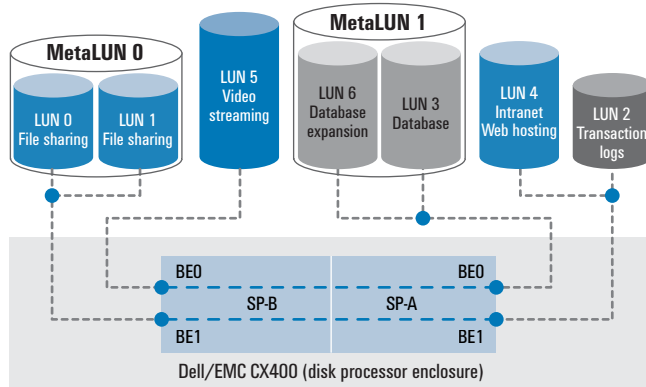


Figure 5. New metaLUNs after expansion for hypothetical small-business scenario

department by performing a stripe expansion of its database, doubling database capacity while maintaining current system performance levels. Figure 5 shows the new metaLUNs after expansion.

Viewing LUNs in the operating system

LUNs will appear as physical disks in the operating system’s tools (for example, the Windows Disk Management utility). Just as RAID groups can be allocated into a single LUN or multiple LUNs, the LUNs can be partitioned as one mapping or multiple-drive mappings in the operating system. Figure 6 depicts allocation of one partition per LUN (Disk 0 to F:\, Disk 1 to G:\, and Disk 2 to H:\) as well as multiple partitions per LUN (Disk 1 to F:\, G:\, and H:\). Again, having a single partition or multiple partitions per LUN depends on the workload for which the partition is being designed.

Optimizing SANs for enhanced storage performance

This two-part article series has explored how SAN performance is affected by RAID configuration, LUN design, and SP cache settings as well as the interrelationships among LUNs, RAID, and back-end loops. Tuning and optimizing these factors can help to build SANs that achieve optimal I/O performance and low latency. The best practices presented in these articles can help provide a solid foundation for excellent storage performance, but should be heeded in relation to the needs and workload requirements of each storage environment. By considering application-specific performance, capacity, and redundancy needs, administrators can build an optimal storage solution.

Arrian Mehis is a systems engineer on the Server and Storage Performance team in the Dell Enterprise Product Group. Arrian has a B.S. in Computer Engineering with a minor in Information Systems from the Georgia Institute of Technology.

Scott Stanford is a systems engineer on the Server and Storage Performance team in the Dell Enterprise Product Group. He has an M.S. in Community and Regional Planning from The University of Texas at Austin and a B.S. from Texas A&M University.