EMC SCALEIO DESIGN CONSIDERATIONS AND BEST PRACTICES

ABSTRACT

This white paper provides technical information, best practices, and hardware and software design considerations for planning the deployment of ScaleIO

June, 2016

To learn more about how EMC products, services, and solutions can help solve your business and IT challenges, contact your local representative or authorized reseller, visit www.emc.com, or explore and compare products in the EMC Store

Part Number HXXXXX <required, see Part numbers below for more info>

TABLE OF CONTENTS

TABLE OF FIGURES

VERSION HISTORY

Version	Date	Details
1.0	May 2016	Initial release of guide. Covers ScaleIO versions 1.33 and 2.0.
1.1	June 2016	Renamed guide, edited/updated best practices.

INTRODUCTION

ScaleIO is an industry-leading software-defined storage solution that enables customers to extend their existing virtual infrastructure into a high-performing virtual SAN. All the hosts (real or virtual), with direct attached storage (DAS) can be pooled together in a single storage system such that all the servers participate in servicing the I/O requests using massive parallel processing. ScaleIO can scale from as little as three hosts to just over a thousand. Capacity and throughput can be increased or decreased on the fly by adding or removing hosts, with no impact to applications or users and no disruption to operations.

The goal of this white paper is to provide best practices for ScaleIO deployment. This white paper also describes performance tunings that should be applied to achieve the optimal performance for different workloads. This guide is intended to provide details on:

- ScaleIO deployment best practices
- ScaleIO deployment best practices during manual installation
- Performance tunings for optimal performance

This paper does not intend to provide an overview of ScaleIO architecture. Please refer to EMC ScaleIO Architecture for further details.

AUDIENCE

The white paper's target audience is anyone intending or considering designing and deploying a ScaleIO System.

It is assumed that the reader has an understanding and working knowledge of the following:

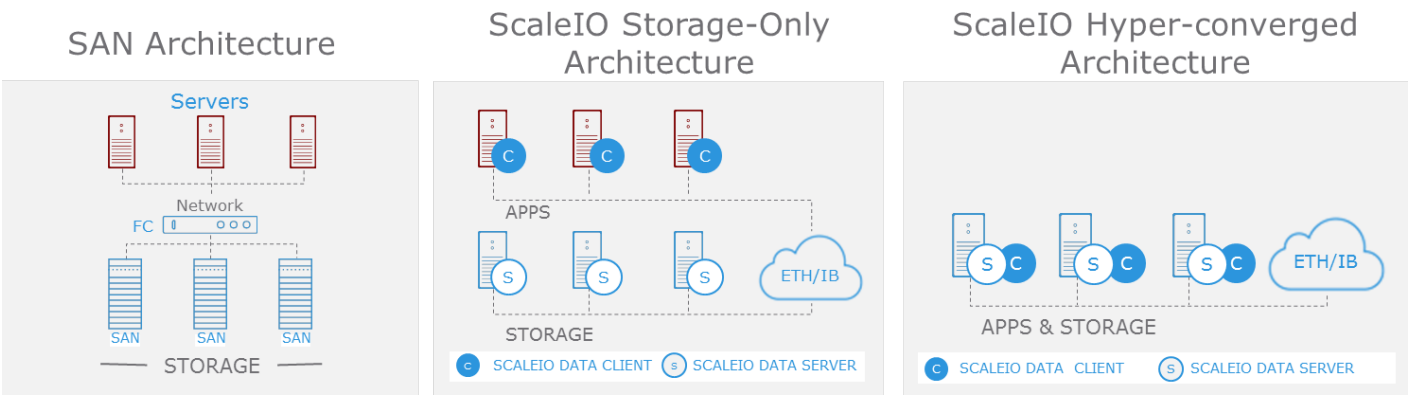
- ScaleIO components, architecture, commands and features
- VMware components, architecture, commands and features

SCALEIO OVERVIEW

The management of large-scale, rapidly growing infrastructures is a constant challenge for many data center operation teams. It is not surprising that data storage is at the heart of these challenges. The traditional dedicated SAN and dedicated workloads cannot always provide the scale and flexibility needed. A storage array can't borrow capacity from another SAN if demand increases. A storage array can also lead to data bottlenecks and a single point of failure. When delivering Infrastructure-as-a-Service (IaaS) or high performance applications, delays in response are simply not acceptable to customers or users.

EMC® ScaleIO® is software that creates a server-based SAN from local application server storage to deliver flexible and scalable performance and capacity on demand. It converges storage and compute resources of commodity hardware into a single-layer architecture, aggregating capacity and performance, simplifying management, and scaling to thousands of nodes. As an alternative to a traditional SAN infrastructure, ScaleIO combines HDDs, SSDs, and PCIe flash cards to create a virtual pool of block storage with varying performance tiers. In addition, it provides enterprise-grade data protection, multi-tenant capabilities, and add-on enterprise features such as QoS, thin provisioning, and snapshots. ScaleIO is hardware-agnostic, supports physical and/or virtual application servers, and has been proven to deliver significant TCO savings vs. traditional SAN.

Figure 1) Traditional storage vs. ScaleIO



Massive Scale - ScaleIO is designed to massively scale from three to thousands of nodes. Unlike most traditional storage systems, as the number of storage devices grows, so do throughput and IOPS. The scalability of performance is linear with regard to the growth of the deployment. Whenever the need arises, additional storage and compute resources (i.e., additional servers and/or drives) can be added modularly so that resources can grow individually or together to maintain balance. Storage growth is therefore always automatically aligned with application needs.

Extreme Performance - Every server in the ScaleIO system is used in the processing of I/O operations, making all I/O and throughput accessible to any application within the system. Such massive I/O parallelism eliminates bottlenecks. Throughput and IOPS scale in direct proportion to the number of servers and local storage devices added to the system, improving cost/performance rates with growth. Performance optimization is automatic; whenever rebuilds and rebalances are needed, they occur in the background with minimal or no impact to applications and users.

Compelling Economics - As opposed to traditional Fibre Channel SANs, ScaleIO has no requirement for a Fibre Channel fabric between the servers and the storage and no dedicated components like HBAs. There are no "forklift" upgrades for end-of-life hardware. You simply remove failed disks or outdated servers from the system. It creates a software-defined storage environment that allows users to exploit the unused local storage capacity in any server. Thus ScaleIO can reduce the cost and complexity of the solution resulting in typically greater than 60 percent TCO savings vs. traditional SAN.

Unparalleled Flexibility - ScaleIO provides flexible deployment options. With ScaleIO, you are provided with two deployment options. The first option is called "two-layer" storage-only and is when the application and storage are installed in separate servers in the ScaleIO system. This provides efficient parallelism and no single points of failure. The second option is called "hyper-converged" and is when the application and storage are installed on the same servers in the ScaleIO system. This creates a single-layer architecture and provides the lowest footprint and cost profile. ScaleIO provides unmatched choice for these deployments options. ScaleIO is infrastructure agnostic making it a true software-defined storage product. It can be used with mixed server brands, operating systems (physical and virtual), and storage media types (HDDs, SSDs, and PCIe flash cards). In addition, customers can also use OpenStack commodity hardware for storage and compute nodes.

Supreme Elasticity - With ScaleIO, storage and compute resources can be increased or decreased whenever the need arises. The system automatically rebalances data "on the fly" with no downtime. Additions and removals can be done in small or large increments. No capacity planning or complex reconfiguration due to interoperability constraints is required, which reduces complexity and cost. The ScaleIO system reconfigures itself as the underlying resources change; data is rearranged and spread evenly on the servers to optimize performance and enhance resilience. All of this happens automatically without operator intervention and therefore eliminates the need for costly and disruptive data migrations.

Essential Features for Enterprises and Service Providers - ScaleIO offers a set of features that gives you complete control over performance, capacity and data location. For both private cloud data centers and service providers, these features enhance system control and manageability—ensuring that quality of service (QoS) is met. With ScaleIO, you can limit the amount of performance—IOPS or bandwidth—that selected customers can consume. The limiter allows for resource distribution to be imposed and regulated, preventing application "hogging" scenarios. Data masking can be used to provide added security for sensitive customer data. ScaleIO offers instantaneous, writeable snapshots for data backups.

For improved read performance, DRAM caching enables you to improve read access by using SDS server RAM. Fault sets – a group of SDS that are likely to go down together, such as SDSs residing on nodes in the same physical rack – can be defined to ensure data mirroring occurs outside the group, improving business continuity. You can create volumes with thin provisioning, providing on-demand storage as well as faster setup and startup times.

ScaleIO also provides multi-tenant capabilities via protection domains and storage pools. Protection domains allow you to isolate specific servers and data sets. This can be done at the granularity of a single customer so that each customer can be under a different SLA. Storage pools can be used for further data segregation, tiering, and performance management. For example, data that is accessed very frequently can be stored in a flash-only storage pool for the lowest latency, while less frequently accessed data can be stored in a low-cost, high-capacity pool of spinning disks.

SCALEIO ARCHITECTURE

A ScaleIO storage system consists of multiple instances of at least three main components working together in a clustered configuration:

- ScaleIO Data Client (SDC)

A lightweight block device driver that exposes the ScaleIO shared volumes to the application. The SDC runs on the same server as the application.

For VMware deployments, SDC is installed directly on the ESX server and provides LUNs to be used as guest storage.

- ScaleIO Data Server (SDS)

  A lightweight software component that is installed on each server which contributes to the ScaleIO storage. SDS interfaces with the host's volume manager and owns the local storage that contributes to the ScaleIO storage pools. The role of the SDS is to perform the back-end IO operations as requested by an SDC.

  For VMware deployments, SDS is installed by deploying a preconfigured virtual machine known as the ScaleIO Virtual Machine (SVM).

- Metadata Manager (MDM) and Tie-Breaker (TB)

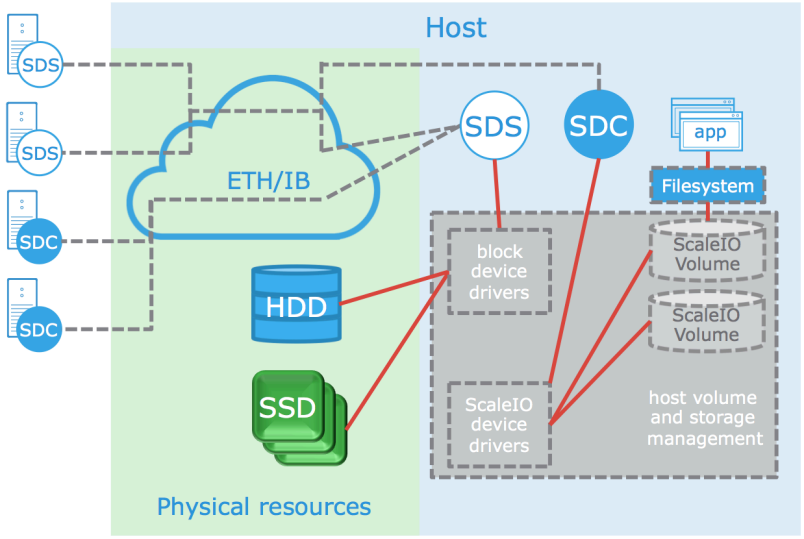
  A repository of storage metadata that manages the ScaleIO system and contains data required for system operation. MDM manages metadata such as device mappings, volume information, snapshots, device allocations, used and remaining capacity, errors and failures. The MDM also monitors the state of the storage system and is responsible for initiating system rebuilds and rebalances. MDM interacts asychronously with SDC and SDS using a separate data path and will not impact their performance.

  Every ScaleIO storage system has a minimum of 3 MDMs: a Master MDM (active), one or more Slave MDMs (passive) and a Tie-Breaker (TB) to resolve any 'split-brain' I/O scenarios . MDM uses an Active/Passive methodology with a Tie Breaker component where the master MDM is always active and the slaves are passive. In ScaleIO 2.0 a deployment can be configured with a total of 5 MDMs (3 MDMs and 2 TBs).

  As with the SDS, in a VMware deployment the MDM and TB are installed with the ScaleIO Virtual Machine.

In a hyper-converged implementation, SDS, SDC, and customer application all coexist on the same compute node. Direct Attached Storage (DAS) is consumed by the ScaleIO SDS and added to the ScaleIO system Storage Pool. Figure 2 shows an example configuration where a majority of the local disk storage is allocated to ScaleIO.

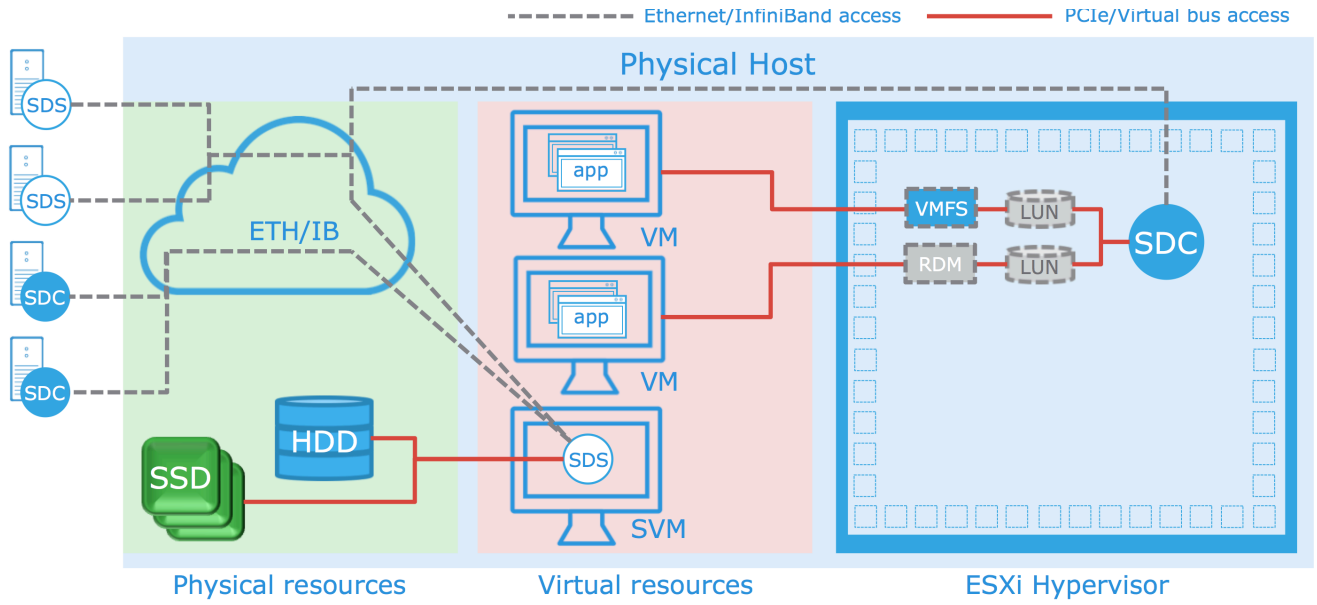
Figure 2) ScaleIO host architecture



For VMware implementation, MDM/TB is installed in the ScaleIO Virtual Machine (SVM).

As mentioned before, SDS and MDM/TB are installed in a ScaleIO Virtual Machine (SVM) and SDC is installed directly on the ESX server (or even potentially on a guest operating system). The LUNs in the figure above can be formatted with VMFS and then exposed using the ESXi hypervisor to the virtual machine, or they can be used as an RDM device. When the LUNs are used as an RDM device, the VMFS layer is omitted, and the guest operating system applies its own filesystem or structure to the presented volume. Figure 3 shows the ScaleIO configuration for a VMware system.

Figure 3) ScaleIO ESXi host architecture

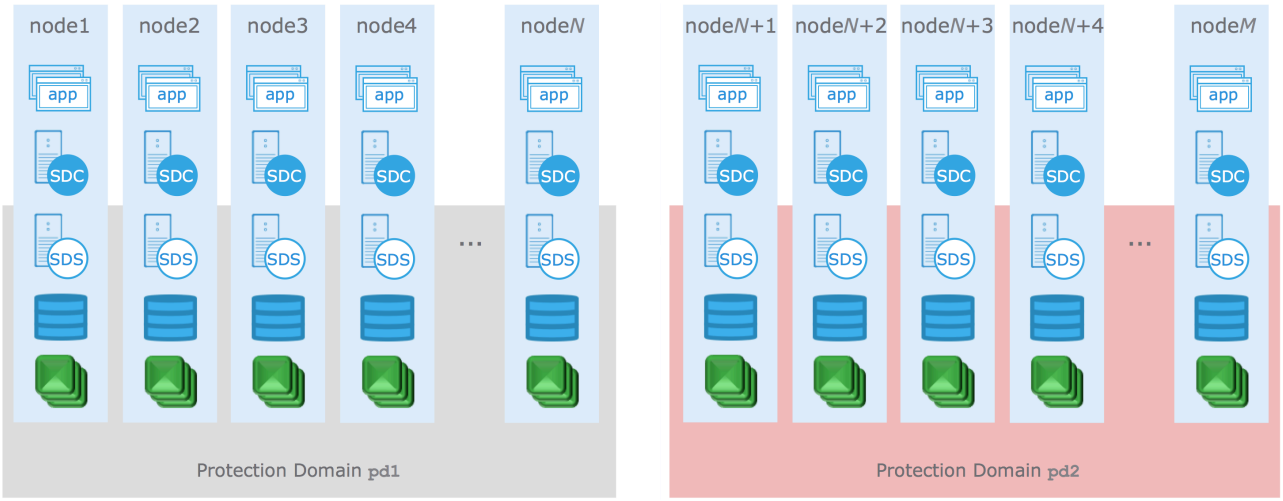


CONCEPTS AND DESIGN CONSIDERATIONS

PROTECTION DOMAINS

A Protection Domain is a set of SDSs configured in separate logical groups. These logical groups allow a storage administrator to physically and/or logically isolate specific data sets and performance capabilities within specified protection domains and limit the effect of any specific device failure. An SDS may belong to only a single Protection Domain, so they may be used to divide a ScaleIO system into smaller logical systems. This is useful in an environment where departmental segregation is desired as well as providing a way to allocate different levels of performance for different clients. Volumes are defined within a Storage Pool belonging to a Protection Domain. A ScaleIO system may have many Protection Domains.

Figure 4) ScaleIO Protection Domain



Pictured above, two Protection Domains, containing a group of SDSs.

Protection Domains can provide:

- Increased resilience: toleration of simultaneous failures in large systems, provided each failure is in a separate Protection Domain. When this is the case, a ScaleIO system can tolerate multiple drive failures in different Protection Domains comprised of different host servers, since each is effectively a separate smaller system: their interdependence is limited to the MDM and Tie-Breaker instances and whatever network infrastructure the Protection Domains share.

- Performance isolation when needed: Different Protection Domains can provide varying levels of performance to clients by having independent groups of SDS nodes with different media performance attributes serving different sets of clients. A high performance Protection Domain could be configured separately from a high capacity Protection Domain.

- Data location control (e.g. multi-tenancy or departmental partitioning): For greater security and isolation, each department or tenant could be assigned a separate Protection Domain. This would prevent one set of clients from impacting a different Protection Domain.

- Physical grouping to help to fit network constraints: Different sets of SDS nodes could be grouped into separate Protection Domains to accommodate network requirements, i.e. additional distinct subnets could each be assigned a unique Protection Domain.

- Higher MTBF with multiple Protection Domains: By sub-dividing a ScaleIO system into Protection Domains, reliability can be increased.

Protection Domains also provide a means to partition a larger system into smaller units, providing for increased protection against simultaneous failures across the overall deployment and increased MTTDL (Mean Time To Data Loss). This may seem counterintuitive, but as the number of devices increases, the probability of simultaneous failure also increases, causing data availability to go down. Consider that the likelihood of a drive failing during a rebuild operation increases as the number of drives increases. The larger the set of devices, the higher probability that simultaneous or during-reconstruction drive failure can occur. It is therefore recommended to not exceed 128 nodes per Protection Domain for a typical ScaleIO deployment. The optimal number of nodes per Protection Domain can be determined by the number and type of nodes (i.e. number of HDD or SSD drives per node) and number of disks.

Departmental segregation is easily addressed by implementing multiple protection domains. Departments (like tenants) can be segregated efficiently and securely; one client will not affect another and performance is effectively segregated. At the same time, storage administration is less time-consuming compared to implementing multiple independent ScaleIO systems. Carefully consider and plan Protection Domain implementation prior to system provisioning.

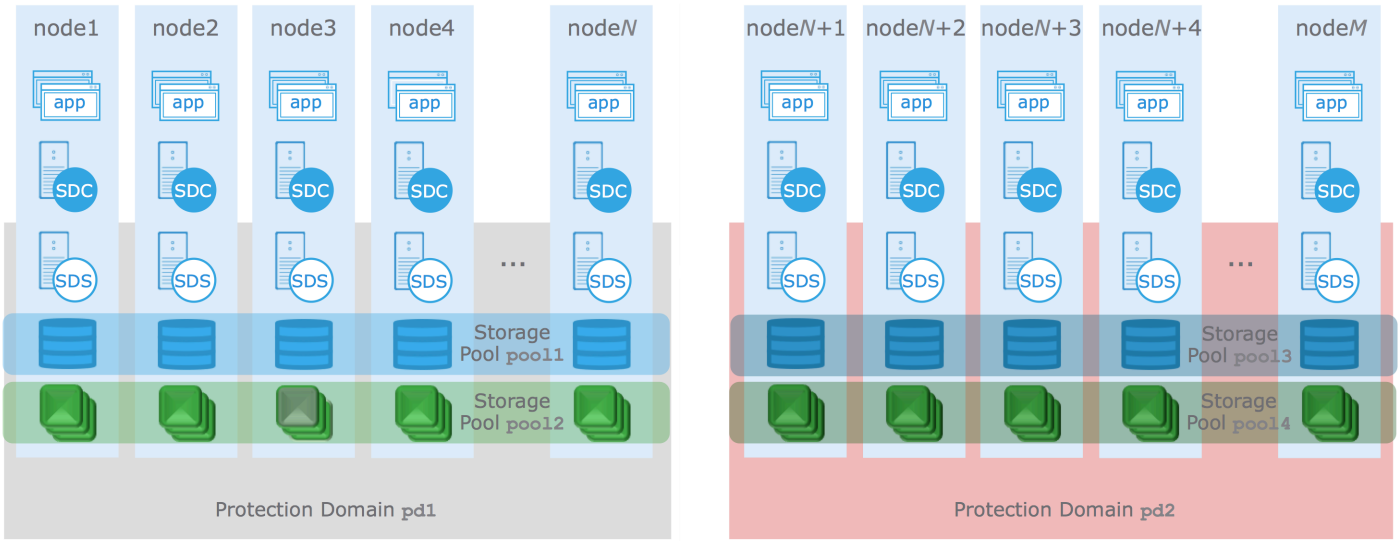
STORAGE POOLS

A Storage Pool is a set of media devices within a Protection Domain used to separate different types and numbers of disks (either HDD, SSD, or PCIe I/O acceleration cards) into pools of storage with self-consistent latency and throughput characteristics. For this reason, you should not mix different types of devices within the same storage pool. For example, if an SDS has both an HDD and an SSD, the HDD and SSD should be in separate storage pools. Similarly, an SDS with multiple HDD devices may provision them in the same storage pool. However, it isn't mandatory that multiple HDDs or SSDs in the same SDS belong to the same storage pools; an SDS with two HDDs could place each in a different storage pool, for instance. Storage Pools act as containers for the volumes presented to SDCs.

A volume for an SDC is defined from a single Storage Pool.

Storage Pools are useful when there are multiple storage hardware types available. Multi-tiering is where a group of magnetic disks (HDD) and Flash (SSD) existing within the same SDS are each configured into separate Storage Pools. Each will have distinct performance characteristics and it is beneficial to separate them into different pools of storage for better performance.

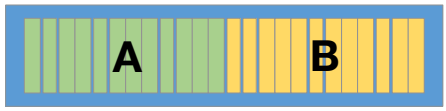
Figure 5) ScaleIO Storage Pools



In the above example, pool1 consists of three HDDs from each SDS, in Protection Domain pd1, pool3 consists of three HDDs from each SDS in Protection Domain pd2, and pool2 and pool4 consist entirely of SSD devices in their respective Protection Domains. Each pool will have different performance capabilities. This storage pool arrangement differs in comparison to using SSD drives for caching (such as with XtremCache or other host-side caching solution): these use the SSD devices to provide a layer of temporary cache for "hot" data between clients and HDD storage, whereas a Storage Pool is a non-shared, separate storage instance. More spindles usually equates to higher performance and in the example above, if all the HDD drives were identical, pool2 would deliver a higher performance than pool1. However, pool3 (consisting of SSDs) would have the highest storage performance; they are not used for cache.

Consider typical nodes with many drives; provisioning multiple storage pools per node allows for easy expansion of the pools as more nodes are added and provides additional protection against multiple failures, as in Figure 6.

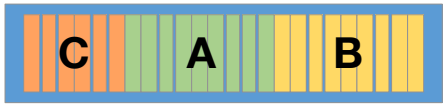
Figure 6) Storage Pool example with multiple drives in a node



Pool "A" contains half of the drives of the node and pool "B" contains the other half.

In Figure 7, the pools of SSD drives are provisioned in pool "C" while pools "A" and "B" consist of the remaining HDDs.

Figure 7) Storage pool with both SSD and HDD



Having a defined number of predetermined Storage Pool types can help with scalability. You can maintain similar performance between the pools by using the same type of device and the same device capacity (or very close). This doesn't just go for the devices being used for storage, but also for other system characteristics like the quantity of RAM available to be used for a read cache. This also provides a means to expand the system by adding new nodes and dividing up the new disks amongst the existing storage pools, instead of provisioning a new storage pool with a much smaller number of disks that would be unable to provide the same level of performance.

Storage Pools can be configured in a number of ways, and depending on what the specific customer requirements are, can provide additional performance, protection, and room for expansion.

STORAGE POOL SIZING

Larger Storage Pools provide:

- Wider striping by spreading I/O across more devices. This benefits applications with high I/O concurrency and bandwidth requirements because more spindles generally equals more throughput and higher performance.

- More flexible shared capacity for multiple volumes by providing a larger pool for volume allocation. ScaleIO is elastic, and additional resources can be added up to 300 drives per storage pool.

- Faster rebuild time. By involving higher numbers of spindles, rebuild times are reduced in most cases.

Smaller Storage Pools provide:

- Better protection from multiple drive failures; less hardware generally equates to higher availability.

- Better guaranteed IOPS by provisioning dedicated Storage Pools for those applications requiring guaranteed performance. Storage pools are also isolated from one another in terms of performance, so contention between applications for storage resources are minimized.

An obvious use case for Storage Pools is to provide a level of performance for different types of hardware (SSD and HDD) corresponding to client contracted SLAs. One Storage Pool could be configured with only SSDs and another with HDDs. Each pool will have different performance characteristics (SSD for speed, HDD for capacity), in addition to having the ability to throttle IOPS by Volume. Typically, cache configurations are also provisioned consistently across storage pools, providing uniform performance from each SDS in the pool.

Multi-departmental tenancy can also be implemented with Storage Pools if the requirement for separation is at a volume level. However, Protection Domains provide physical separation while Storage Pools provide logical; SDCs can access volumes regardless of Protection Domain or Storage Pool in the same manner. True multi-tenancy is best implemented via separate discreet ScaleIO systems. Because Storage Pools can be expanded up to a maximum of 300 devices, carefully plan out the provisioning of them prior to system deployment to provide for expansion and adequate performance.

FAULT UNITS AND FAULT SETS

ScaleIO contains the concept of fault units when considering SDSs participating in a storage pool. In order to make sure that all data is redundantly mirrored, ScaleIO ensures that a primary and secondary copy of data will not reside within the same fault unit. For a default installation, a fault unit is equivalent to any given individual SDS within the ScaleIO cluster, and so ScaleIO ensures that primary and secondary copies of data never reside on the same host.

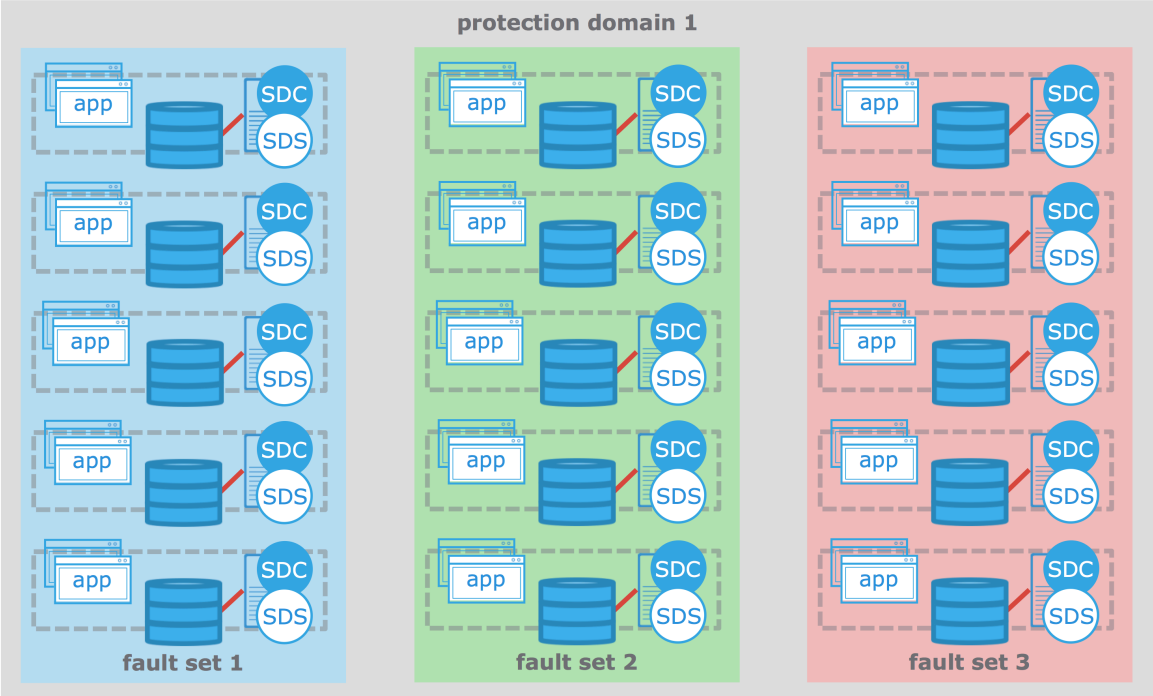
There are some cases where deploying a ScaleIO in a configuration where a fault unit is larger than the default of one SDS makes sense, and in this case a Protection Domain can be divided into Fault Sets.

NOTE: You can only create and configure Fault Sets before adding SDSs to the system. Configuring them incorrectly may prevent the creation of volumes, since an SDS can only be added to a Fault Set during the creation of the SDS. Small implementations generally perform best with the default "none" Fault Set configuration..

Fault Sets ultimately determine how a failure can be tolerated. ScaleIO Fault Sets are subgroup of SDSs installed on host servers within a Protection Domain. They define where the actual copies of data exist in a ScaleIO system; each copy (Primary or Secondary) resides in a different Fault Set. Data mirroring for all devices in the Fault Set must take place on SDSs in a different Fault Set. A minimum of three Fault Sets must be defined; ScaleIO provides additional protection vs. RAID 1 by allowing for a failure of a disk (or node) and still providing protection with two copies of user data, hence the need for a minimum of three Fault Sets. Thus, no two copies (Primary or Secondary) of data will reside on the same Fault Set. When defining Fault Sets the term *fault unit* is used to express a single unit of failure. Fault units can be a Fault Set or a single SDS not defined within an existing Fault Set (essentially a Fault Set of a single SDS).

Fault Sets allow grouping of SDSs that are likely to experience a failure together. Consider a multi-node chassis with four blades, configured for four SDS units. If the chassis is down, all four SDS units are down. Therefore, it would be advantageous to configure all four nodes as a single Fault Set to increase availability for exactly this scenario. This ensures that for each data block on each SDS unit in the chassis, the corresponding copy of the data block is in a different Fault Set, and consequently a different chassis not subject to failure of this single blade or of the entire multi-blade chassis. Another use case for configuring multiple nodes into a single fault set would be a rack containing several SDSs, as seen in Figure 8.

Figure 8) Fault Sets defined on a per-rack basis



Each rack is configured as a Fault Set, and so the loss of any single rack will still allow for a complete rebuild of all data; any single rack failure will not cause data loss.

Fault Sets are NOT a substitute for a disaster recovery (DR) plan. Rather, they provide a method of increasing data availability and continuous operations by spreading drive/node risk throughout a system and allowing some guarantees to be made about data locality amongst a Protection Domain's member systems.

SIZING USABLE STORAGE AND SPARE CAPACITY

Since ScaleIO is designed for enterprise use, redundancy is built into its architecture and all data is protected by storing multiple copies. ScaleIO uses RAID 1 mesh-mirrored layout to protect user and application data. Each piece of data is randomly distributed across fault sets and SDSs, resulting in two copies for data availability. Because of this, usable capacity is calculated by accounting for the two copies of the data and reserving some spare storage. Spare capacity should be as big as the largest Fault Set.

Sizing usable storage in a system can be calculated using the following data:

- Number of servers: n
- Capacity per server: c terabytes (TB)
- Number of server failures to tolerate: k
- Raw storage minus spare capacity: $(n - k) * c$
- Usable storage: S

The formula is:

$$S = \frac{(n-k)*c}{2}$$

A simple example with 11 servers, 3TB each, tolerating 1 server failure:

$$S = \frac{(11-1)*3}{2} = 15 \text{ TB}$$

Therefore, 15TB of usable space is available from 33TB raw storage (including 3TB spare capacity for one server failure).

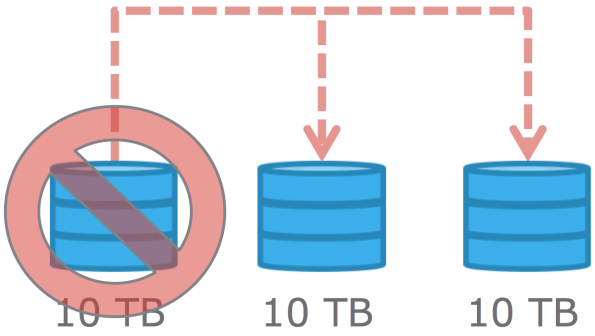
Recall that in a ScaleIO system, Fault Sets are a group of SDSs within a Protection Domain.

With fewer fault sets, the amount of spare capacity required is increased when compared to a system of many fault sets.

EXAMPLE 1

Consider the bare minimum of three fault sets, each with 10TB. In order to rebuild a single fault set failure, the two remaining fault sets must have between them spare capacity equal to the size of the failed fault set. Half of the remaining storage must be reserved to rebuild a failed fault set, as seen in Figure 9 with a raw system size of 30TB.

Figure 9) Spare capacity with three fault units



If one of the SDS nodes fails, the data contained within must be re-protected on the remaining nodes. The 20TB remaining must be divided in half to allow for mirroring of the data, thus leaving 10TB usable on a 30TB three node system. This configuration requires tremendous overhead, and so a typical deployment will define more than three fault sets. The usable storage calculation is as follows:

$$S = \frac{(3-1)*10}{2} = 10 \text{ TB usable}$$

From this you can see the way that ScaleIO becomes more storage efficient as more nodes are added and the required spare capacity can be evenly distributed.

EXAMPLE 2

Consider three racks of ScaleIO nodes, each with 20 nodes of 2TB each yielding a total raw capacity of 120TB. The customer wants to protect against a rack failure (20 nodes). The calculation is as follows (where k=20):

$$S = \frac{(60-20)*2}{2} = 40 \text{ TB usable}$$

As the number of fault sets increases, the reserved overhead decreases. Compare the same deployment when the system does not require an entire rack as the largest unit of failure (k=1):

$$S = \frac{(60-1)*2}{2} = 59 \text{ TB usable}$$

So you must carefully consider the pros and cons of fault unit size and choose the best implementation to meet your protection requirements.

- Spare capacity should be configured a minimum of $\frac{1}{N}$, where N is the number of nodes in the system.
    - For a ten node system it should be set to $\frac{1}{10}$ or 10%.
    - For a three node system, the recommended setting would be $\frac{1}{3}$ or 33%.
- Note that this recommendation for spare capacity applies equally to fault sets: if you have three fault sets, then spare capacity should also be $\frac{1}{3}$ or 33%.
- Spare capacity is implemented at the Storage Pool level; each Storage Pool will have a separately provisioned spare capacity.

- Consider system free space in addition to spare capacity; in the case of sequential recovered failures where the amount of spare capacity is reduced to zero, a system at or near full capacity (i.e. 99%) will have no additional free space or spare capacity to rebuild in case of failure.

It is better to err on the side of caution and set the spare pool to be larger than strictly necessary if the customer is concerned with reliability and resiliency, since the system always requires adequate spare capacity to rebuild. Ensure that the spare capacity is at least equal to the amount of capacity in the node containing the maximum capacity or the maximum Fault Set capacity.

SYSTEM BANDWIDTH GUIDELINES

When a ScaleIO storage system is performing rebuild and/or rebalance operations, it may be beneficial to limit the rebuild/rebalance bandwidth of your ScaleIO system. Since ordinary read/write operations also involve intersystem communication by SDC nodes to SDS nodes (as well as SDS to SDS communication for the copy), rebuild and/or rebalance operations can affect the amount of available bandwidth for client traffic if the physical ports are shared between SDC nodes and SDS nodes. ScaleIO systems can allocate bandwidth to clients (front-end) and internally between SDS nodes (back-end) while rebuilding or rebalancing.

By default, the amount of bandwidth that a ScaleIO system can deliver is directly related to the number of interfaces available per node throughout the system. Back-end system operations such as rebuilding and rebalancing can have an adverse effect on performance when they are active and competing for bandwidth over a single interface. Multiple interfaces are always recommended, allowing for the front-end (client) traffic to flow unimpeded by back-end operations.

Client (SDC) bandwidth can be limited by restricting maximum IOPS or bandwidth in MB/sec. For example, the following command will limit the IOPS available to a specific client (192.168.1.3) to a specific volume to 100:

```
scli --mdm_ip 192.168.1.200 --set_sdc_volume_limit --volume_name vol1 --sdc_ip 192.168.1.3
--limit_iops 100
```

If a ScaleIO SDS node has only a single 10Gb interface, it may be beneficial to limit the client bandwidth to keep many clients from overloading the system while a rebuild or rebalance occurs. Best practice is to implement a system with a minimum of two 10GbE interfaces to provide for adequate bandwidth between nodes while a rebuild or rebalance is in progress. Additional interfaces provide greater bandwidth, and allow intensive back-end operations such as rebuild and rebalance to progress without competing with front-side client traffic. Interfaces can be designated to for use as SDC to SDS communication or SDS to SDS traffic only

REBUILD THROTTLING

Parallel Rebuild Rebalance Jobs: It is recommended to leave this setting at its default value of 2. Raising it beyond 2 can cause contention during rebuild and rebalance jobs. This is set at the Protection Domain and/or Storage Pool level.

REBALANCE THROTTLING

Rebalancing the data throughout the system is important, much like defragmenting a single HDD, and an even distribution of data in a volume will produce better performance. Rebalancing is not needed for data protection purposes. The Rebalance job must run, but can be set at a lower priority via throttling to prevent contention with application IO. By default it is set to "Favor Application I/O" and will not compete with client traffic and this is a recommended best practice. This is also set at the Protection Domain and/or Storage Pool level.

CONCURRENT IO PER DEVICE

Number of Concurrent IOs per device during rebuild/rebalance: It is recommended to leave this at a default of "1" since raising this value can cause contention during rebuild and rebalance jobs.

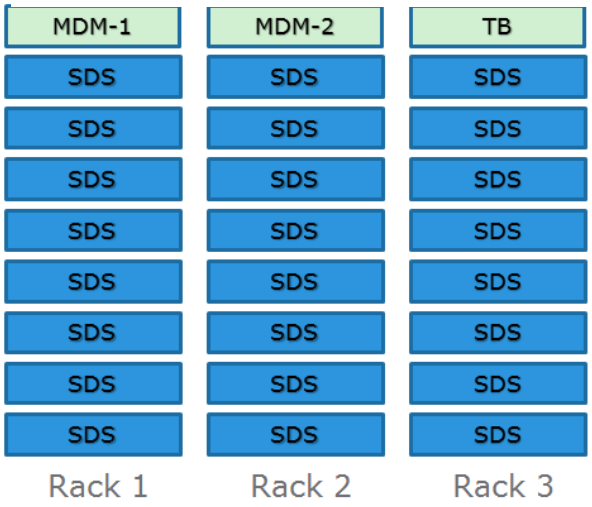
By managing various system related jobs, client performance can be preserved. However, during rebuild, it may be advisable to allow for additional bandwidth, to speed the reprotect process, and once again provide for multiple data copy protection.

MDM AND TIE-BREAKER PLACEMENT

A ScaleIO system's MDM and Tie-Breaker nodes can be located anywhere in a system (and do not require dedicated nodes), but since they are critically important to storage system data availability, in larger configurations, they should ideally be distributed for maximum resiliency. For example, in a multi-rack installation they should placed in different racks. In an installation with multi-node servers, the multi-bladed server should be configured as a fault unit; MDMs should be placed in different fault units and consequently in different physical chassis. See the previous discussion on Fault Sets where a single rack was designed as a fault unit.

If a rack goes down containing a Master MDM (MDM-1), then a Slave MDM (MDM-2) becomes the Master in

Figure 10) MDM and Tie-Breaker placement



Note that the MDMs and Tie Breaker (TB) are each in separate racks, thus minimizing the chance that both MDMs could be down at the same time should a complete rack fail.

In addition, both MDMs and TB nodes must have network connectivity to every SDS node, preferably with the least amount of latency possible. Ideal system designs minimize any sort of routing, and optimally within the same subnet.

NETWORKING

The network connecting ScaleIO components is critical to its overall performance and resilience, and this is the case whether ScaleIO is deployed as part of a 2-layer or hyperconverged infrastructure. A ScaleIO system of any size will want to take steps to maximize the quality and throughput while minimizing the response time of the system as far as the available network infrastructure will allow.

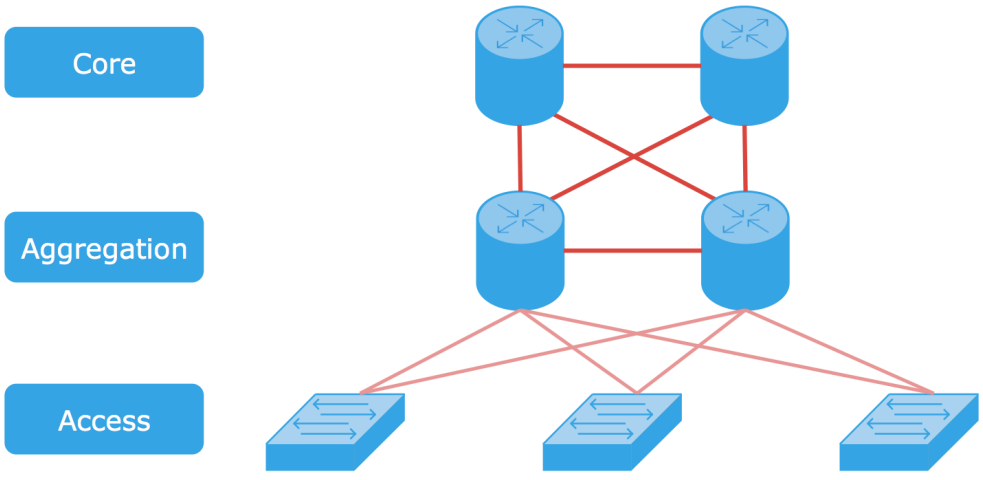
There are three main networking considerations for a ScaleIO implementation:

- The system's physical network topology and the links between nodes
- Logical network architecture: routing between nodes and groups of nodes, segregation of client and storage traffic
- Host-based networking provisioning, interface teaming, and multipathing

When using ScaleIO, every node providing storage is participating in a mesh network with every node consuming that storage. The ideal network topology is therefore one where every node has full and unrestricted access over redundant network links to every other node, since the ScaleIO system as a whole is capable of making use of all network resources available to it.

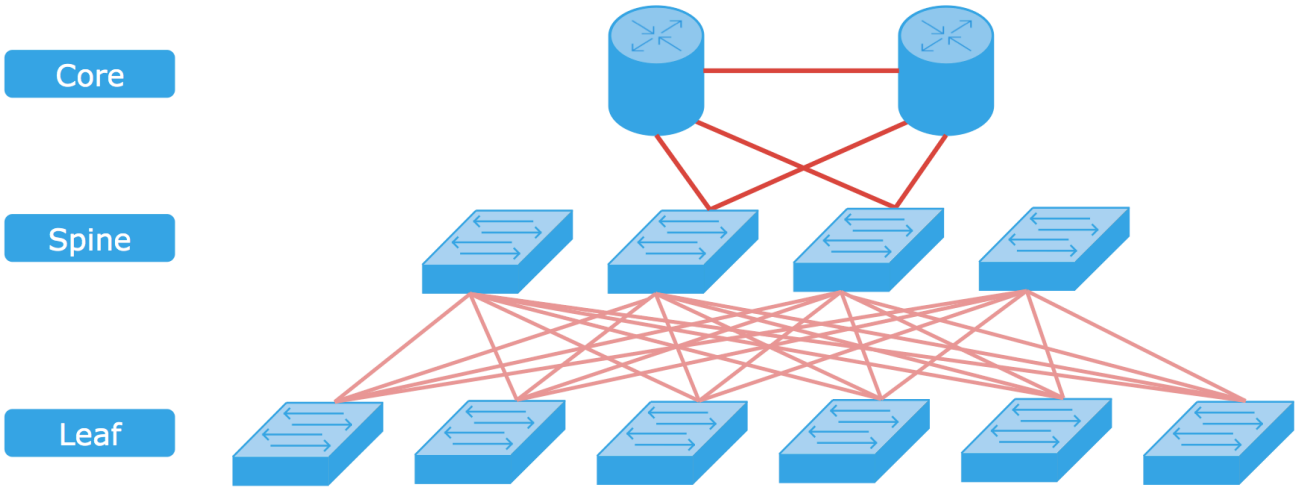
For smaller storage systems, or ones not planning on large increases to system scale, simply running a full-mesh network where SDS and SDC share a non-routed subnet can be an ideal solution with high performance and throughput and low complexity.

Figure 11) Flat network topology



For systems with hundreds of nodes, this approach becomes much more expensive. Instead, a network with a more complex topology would be better suited, along with commonly deployed network management technologies such as link aggregation, advanced routing, and traffic shaping. For instance, rather than a mesh network, a leaf-spine network in which the overall cluster size and workloads help to define the ratio of network ports on leaf switches to the ratio of network ports meant for interconnection between spine switches.

Figure 12) Leaf-spine network topology



A separate and in-depth consideration of each of these technologies can be found in the ScaleIO Networking Best Practices and Design Considerations paper.

NETWORK DESIGN CONSIDERATIONS

The following considerations apply to ScaleIO systems at any scale and should be taken into account for all ScaleIO network infrastructure.

MULTIPATHING

ScaleIO software components such as the SDS and MDM are designed to listen and respond to requests on multiple physical or logical network interfaces. LAGs (Link Aggregation Groups) or NIC teaming at the switch or host level is not necessary to provide link-level redundancy. The SDC is capable of both balancing requests across multiple paths and discarding paths that become unavailable without leading to timeouts or data availability issues. However, since many existing network topologies make extensive use of LAGs either to a single switch or multiple switches and use them as a preferred solution for eliminating single points of failure within the network stack, they are also a supported method of achieving link-level redundancy.

METADATA MANAGER

Nodes running the MDM / Tie-Breaker software share a cluster and are in continuous communication via a heartbeat. In terms of overall system stability and resilience, this traffic should be given priority. On networks with contention or a very high rate of utilization, it may benefit the cluster to assign separate interfaces to nodes that will participate as a Metadata Manager and to connect these nodes via a dedicated private network. Any such network should not, if possible, use link aggregation at the host level. Instead, ScaleIO's path management and cluster heartbeat will allow the MDM and Tie-Breakers to remain in contact.

JUMBO FRAMES

Increasing the maximum transmission unit (MTU), also known as enabling jumbo frames, is a common practice on back-end Ethernet networks used for data access. An MTU of 9000, as compared to the default Ethernet MTU size of 1500, effectively means that a higher proportion of each frame transmitted on the network will contain payload data rather than packet metadata. This corresponds to a fractional increase in throughput.

One implication of turning on jumbo frames is that every piece of equipment in the network link layer must share the same 9000 MTU setting from end to end; equipment that doesn't is likely to experience hard-to-diagnose failures.

NETWORK TESTING

Prior to final deployment, the network should be tested with the ScaleIO SDS network test via the CLI in order to establish baseline performance characteristics. A simple ping test can provide latencies between components, and a more rigorous network test should be executed to test connectivity. For example, a test for every ScaleIO node to be deployed, as well as a total throughput concurrency test from nodes attached to different leaves in a leaf/spine network topology.

Verifying adequate network performance (and consistent network connectivity between ALL SDS, SDC, and MDM/TB components) is critical to avoiding issues with ScaleIO. Without a sound network, performance and reliability cannot be achieved.

Network testing is invoked via the CLI:

```
scli --mdm_ip 192.168.1.200 --start_sds_network_test --sds_ip 192.168.1.4 --parallel_messages 8
--network_test_size_gb 2 --network_test_length_secs 120
```

You will see a response:

```
Network testing successfully started
```

After the command finishes, results can be queried via CLI:

```
scli --mdm_ip 192.168.1.200 --query_sds_network_test_result --sds_ip 192.168.1.4
```

Which will produce results similar to:

```
SDS with IP 192.168.1.4 returned information on 2 SDSs
    SDS 4cf480b000000001 192.168.1.5 bandwidth 1.4 GB (1383 MB) per-second
    SDS 4cf480b100000002 192.168.1.6 bandwidth 1.5 GB (1494 MB) per-second
```

Full parameters for both of these commands can be found in the ScaleIO User Guide. Several SDS nodes should be tested to ensure the network performs equivalently across all components.

SCALEIO INSTALLER AND INITIAL SETUP CONSIDERATIONS

ESX SETTINGS – VMDK

If the ScaleIO system is to be used for VMDK files, then that specific setting during installation should be selected. Otherwise, it will save time to NOT enable VMDK as it adds considerable time (on the order of several hours) to initial volume formatting. The ESX installation may appear "stuck" but in reality it is progressing and may take many hours to complete.

GENERAL – ZERO PADDING

Integrity checking requires that Zero Padding be enabled. Integrity checking is an optional background process that will only run when the system network is relatively quiet. Installation with Zero Padding enabled requires that the initial blocks all be zeroed out; otherwise a deleted or "freed" block may contain unneeded data, but that will not match an actual empty block (one which has been zeroed out).

The background device scanner will run in either device only mode (fix any errors during a read of scanned data by requesting fixed data from the peer SDS), or data comparison mode (both SDSes read their copy of the data and compare). Both settings will incur additional bandwidth requirements on the system and consideration must be given to setting a scanner bandwidth limit to prevent negative system performance. This will also extend the time needed to process deletes, since This incurs an additional write penalty on deletes, since deleted blocks need to be zeroed out.

Zero padding is also required if RecoverPoint is to be deployed with the system.

Since zero padding requires that newly initialized volumes have zeroes written to every usable block, enabling it means that during this setup period there may be a performance impact due to the large number of writes that must first be processed by the system.

OTHER SETTINGS - VMWARE

For additional recommended VMware settings, refer to EMC ScaleIO for VMware Environment Deployment and Performance Best Practices.

HOST STORAGE DEVICES

Since ScaleIO handles mesh mirror operations by requiring primary and secondary copies of data to be stored in separate fault units, no host-level mirroring or RAID setup is necessary. The available disks in a host should be simply assigned to that host's SDS.

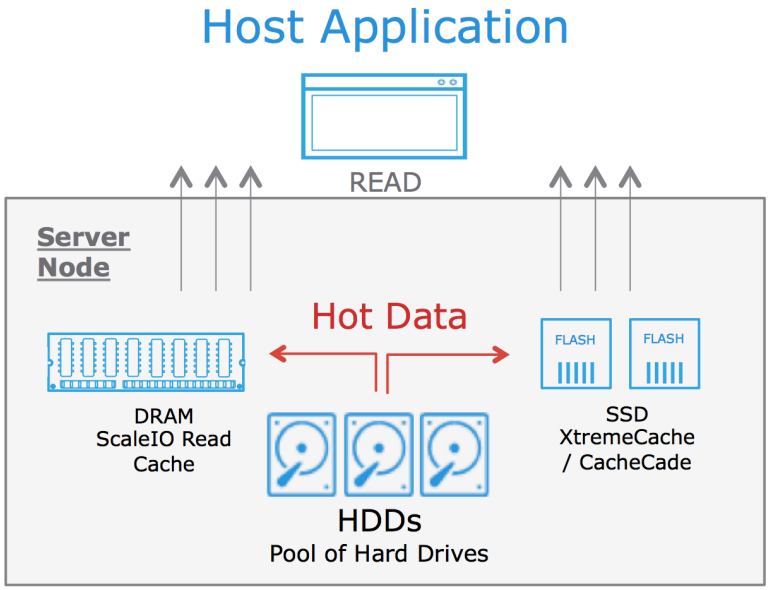
However, hosts that have a write-back cache, whether provided by software or a host storage controller's nonvolatile memory, should be configured to use it in conjunction with ScaleIO.

CACHING STRATEGIES

Caching is a way to maximize the performance of slower devices within a system. Cache can be implemented in several ways on a ScaleIO system.

- RAM Read cache utilizes the RAM memory that is present in the SDS server. RAM Read cache is enabled by default and can be applied at the volume level, Storage Pool level and individual SDSs. Size can be configured at the SDS level from 128MB (default) to 128GB. Up to 50% of the SDS's RAM can be used for cache if the total RAM is less than 32GB; 75% if more than 32GB is present. The same amount of physical RAM should be installed on each SDS node within a Storage Pool.
- RFCache is a read-only caching strategy designed for high-workload systems that have a read-heavy workflow. It does not provide for caching of write-heavy workflows.
- CacheCade is another caching strategy that is built into the RAID controller on a node. It provides both read and write cache by utilizing locally attached SSD storage for caching. Other RAID controllers may offer similar capabilities and can be implemented as well.

Figure 13) Host caching strategies overview



CACHING RECOMMENDATIONS

RAM Read caching may potentially decrease SSD storage performance and is not recommended since SSD read performance is typically very high.

If a server has a hardware RAID controller, ScaleIO prefers to use the controller's caching abilities for better performance. It is best utilized when all of the HDD devices are configured as stand-alone (i.e. setting each of the devices to RAID-0 separately). Furthermore, if the RAID controller is NOT battery-backed, then only READ caching should be enabled to prevent potential data loss in case of power failure. If the RAID controller is battery-backed (for instance, with supercapacitors) then READ/WRITE caching is recommended.

All cache mechanisms are not recommended to be enabled at the same time (i.e. RAM Read cache, XtremCache, and Cachecade should not all be enabled on the same SDS node) as they may use the same caching algorithms and not provide additional benefit when enabled in parallel. However, the RAM read cache may be used with CacheCade potentially benefitting multiple different workflows.

In a hyper-converged implementation, the RAM installed in a node will be shared between SDS, SDC, and application, so consider that RAM used for caching in one area cannot be used in another, and that unlike in a 2-layer ScaleIO implementation, both storage and application are sharing the same pool of RAM; caching at both the client and server level can therefore mean inadvertently duplicating data in cache on the same hardware. Therefore, a caching strategy should be chosen and followed consistently throughout the cluster.

The number of memory buffers that an SDS uses for I/O can also be configured for specific use cases. The more buffers added, the more I/Os the SDS can handle in parallel. This number is derived by the number of devices attached to the SDS. It can be changed to suit the system configuration. Typical values are:

- 1 – Default, which fits lower overall performance (such as HDDs only)
- 3 – Higher performance (such as an SSD environment)
- 5 – For use with the replication splitter for RecoverPoint

For example, if a ScaleIO system is implementing RecoverPoint, then this setting should be changed to "5." Refer to the *Fine-Tuning ScaleIO Performance Technical Notes* before changing this parameter.

Ensure you understand the customer's workflow before deciding on the appropriate size and type of cache.

A "write heavy" workflow will not benefit from read type cache (RAM and XtremCache) thus CacheCade would be an appropriate solution.

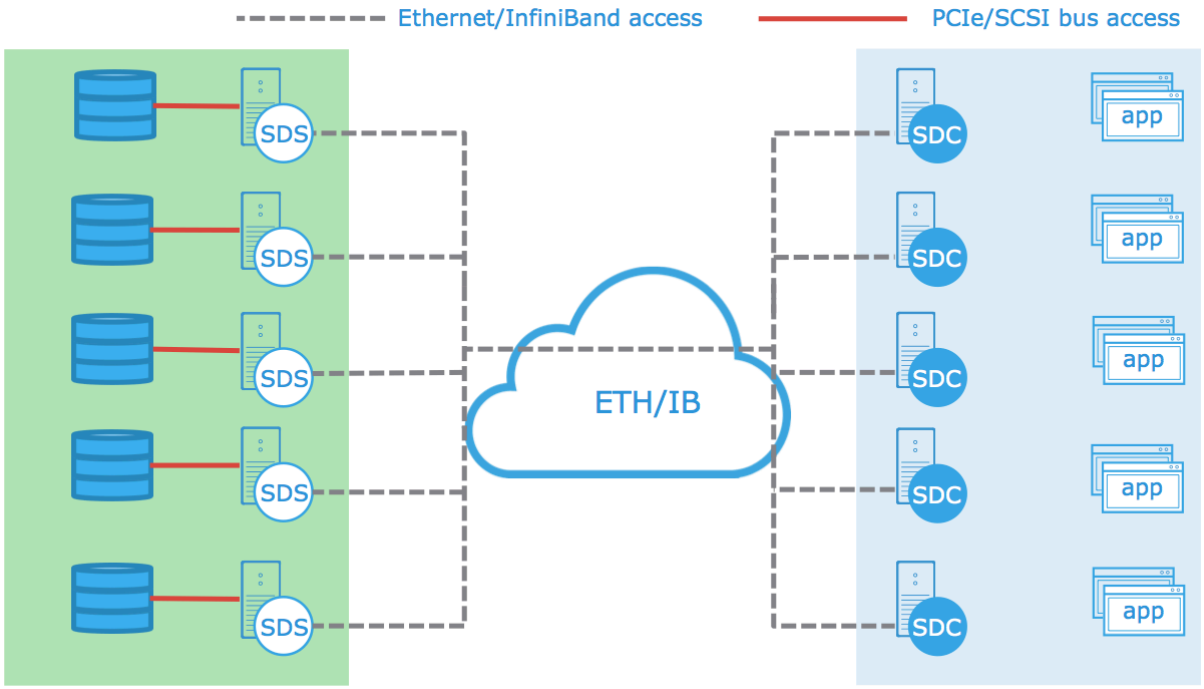
Similarly, a "read heavy" workflow may not need write caching (so use the default RAM Read Cache and add XtremCache if necessary to reach specific performance numbers.

TWO-LAYER, CONVERGED, AND HYPER-CONVERGED DESIGN

ScaleIO system design is extremely flexible. A system can be implemented in a huge variety of ways, from a traditional client-server design to an all-encompassing hyper-converged system where storage, client, and compute exist on every node. SDCs can exist anywhere within a ScaleIO cluster either as stand-alone nodes or coexisting with SDS services in the same node.

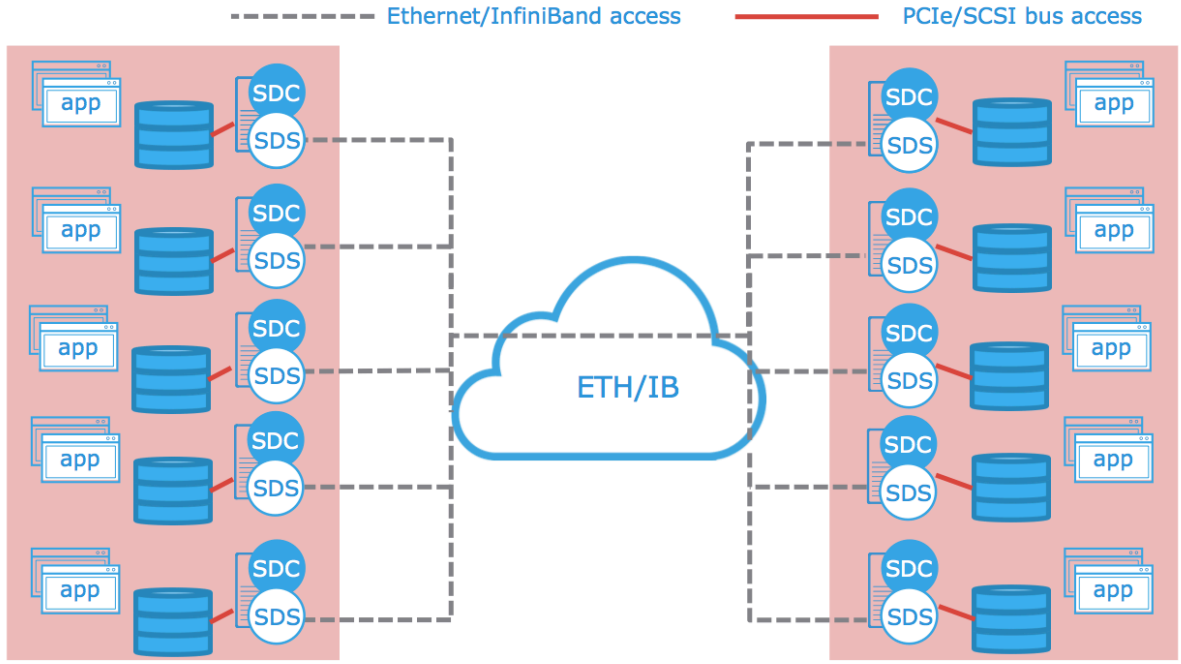
In the following figures, each "C" is an SDC unit, and each "S" is an SDS unit.

Figure 14) Two-Layer architecture implementation



A Two-Layer system provides only storage. The SDS is the only application running on its nodes, and is providing storage for a different set of SDC nodes. This corresponds to traditional SAN deployments as discussed in the introduction.

Figure 15) Hyperconverged architecture implementation



In a hyper-converged system, storage server, storage client and compute (applications) run together within the same node. SDS, SDC, and the application all can run simultaneously on the same hardware, providing a scalable infrastructure.

Given that a ScaleIO system can be configured in a nearly infinite number of ways, these three examples of contemporary system design should provide guidance towards determining what kind of infrastructure is appropriate to match a customer's needs and workflow. ScaleIO can work as a group of independent machines, but given the density and demands that hardware serve multiple purposes at all times, most modern deployments employ convergence in at least some fashion.

Of the three designs presented here, the hyper-converged implementation maximizes utilization of hardware if the applications are not CPU and network intensive.

HOMOGENOUS NODE DESIGN

When designing a ScaleIO system and the logical groups of components within Protection Domains, Storage Pools, and Fault Sets, it should be considered a best practice to provision each SDS identically within a group for the most consistent performance. A non-homogenous group of nodes will mean that performance does not increase in a linear fashion as the group grows, since some nodes may have higher or lower throughput capabilities in comparison to other nodes.

Figure 16) Homogenous SDS nodes

Each of the above SDS nodes has the same quantity (three) of HDD; each with identical size and type, along with a single SSD, each of the same type. All nodes have the same number of network interfaces. For all intents and purposes, each node is identical in hardware configuration.

Each SDS within a Storage Pool should have the same cache settings, same type of disks with like performance, and same number of interfaces so that each SDS node can deliver identical performance. Different pools can of course have different drives (i.e. a pool of SSD, a pool of HDD, and a pool of two HDDs per node are all acceptable configurations, however the types of drives within each pool should be of identical size and configuration for optimal performance.). For example, all disks within a pool should have the same performance characteristics and capacity to deliver the same performance from each node. SSD and HDD devices should not be mixed within the same pool.

Likewise, SDS nodes from different fault sets should also be identical in configuration. RAM, CPU, number and type(s) of disks should be identical in order to provide consistent performance from all SDS nodes.

The system will not be able to deliver consistent performance if different types of components are utilized within the same group.

Though ScaleIO will function with a mixture of different operating systems (OS) implemented in the same system, best performance arises from OS within a Storage Pool. A mixed system of ESX and Linux may lead to unpredictable performance.

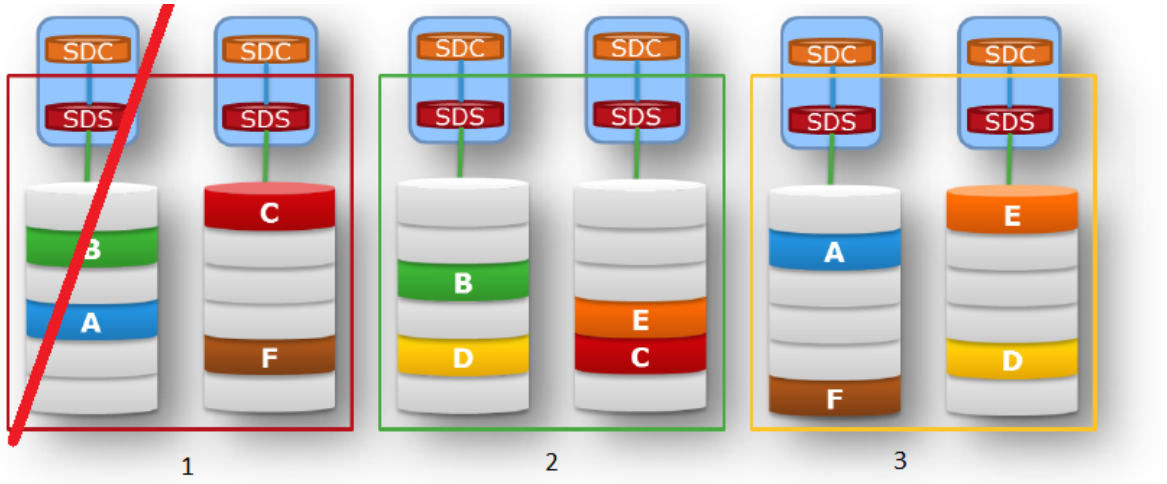
If different components are used, inconsistent performance can occur as some SDS nodes may perform faster than others. Hot spotting within a system may happen where IO requests are unequally divided amongst SDS nodes. Protection Domains may be used to segregate different node types.

RECOVERY SCENARIOS

ScaleIO utilizes a RAID-1 mesh mirrored layout. Each piece of data is stored on two different SDS units. Data is protected by a second copy in a different Fault Set. Since each Fault Set can contain a single copy of a piece of data, data will not be mirrored within a Fault Set but instead across two different Fault Sets and therefore two different SDS nodes. Recall that in a ScaleIO system, Fault Sets are a group of SDSs within a Protection Domain and each SDS can exist in only one Fault Set. Ensuring that a multiple copies of data reside on separate physical storage protects against failure of a single disk, and by extension copies of data across different nodes protects against node failure. By organizing disks and SDS units in different Fault Sets, we can protect against single disk or single node failures.

1. Loss of SDSs in same fault set; data is protected since copies are in different Fault Sets. A single SDS is unavailable is this scenario.

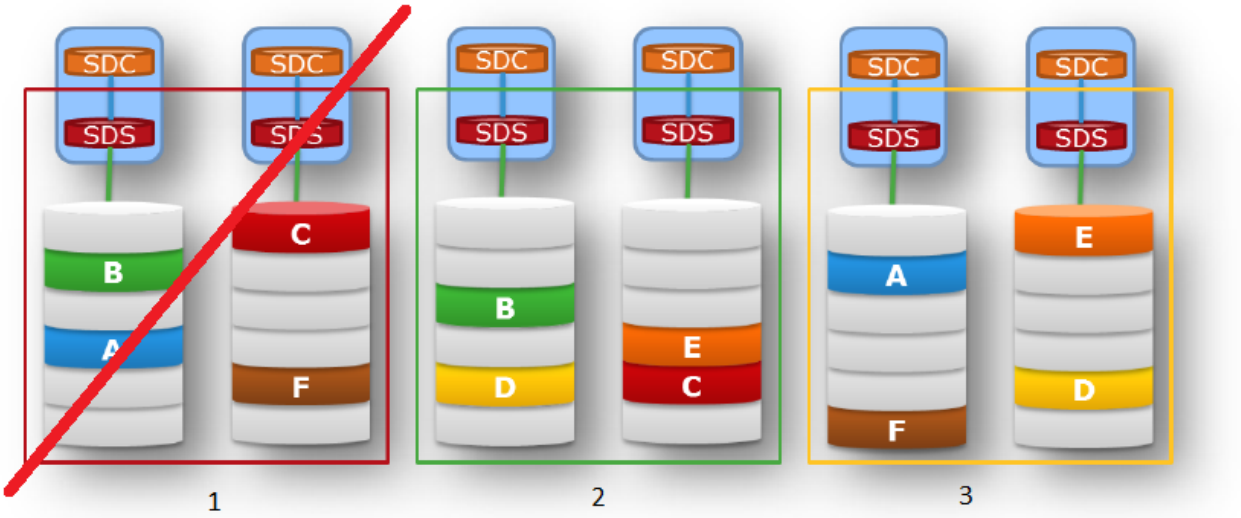
Figure 17) SDS node failure



In the figure above, one SDS is down in Fault Set #1. Blocks in file "A" and "B" are recoverable since copies of their blocks are in different Fault Sets.

2. Loss of fault set; consider the rack example previously discussed where each rack of SDS nodes is a Fault Set. In this scenario, both SDS components are unavailable.

Figure 18) Fault set failure



In this scenario, Fault Set #1 is unavailable (and both SDS components are down), but files "A", "B", "C", and "F" are recoverable since a copy of each of them is still available in the remaining Fault Sets. Rebuild will occur in the remaining Fault Sets and data will again be protected.

In both examples above, no data was lost, and because of the design of the ScaleIO system, data is reprotected with the remaining hardware.

EXAMPLE CUSTOMER DEPLOYMENTS

ScaleIO is a massively scalable architecture and effectively planning a system both for initial deployment and expected growth requires many considerations. In this section, you will see some specific real world examples to demonstrate the flexibility and the resiliency of ScaleIO. By implementing various combinations of Protection Domains, Storage Pools, and Fault Sets, systems can be built to address specific client workflow needs.

DEPLOYMENT I: SMALL SYSTEM

Requirements:

The customer wishes to explore the performance aspects of ScaleIO and desires a small system to start with. A minimal number of nodes (half-dozen) and a few HDD devices are available per node. No specific performance guidelines are documented, but the customer is interested to see what can be accomplished given the hardware available.

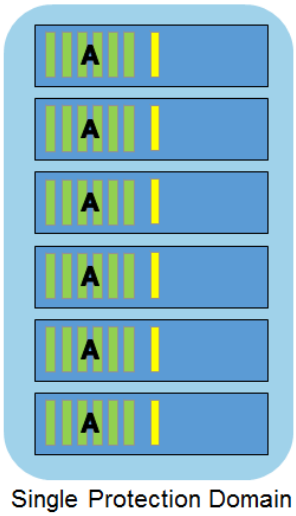
Design Recommendations:

Recall that a ScaleIO system requires a *minimum* of three nodes. In order to maximize performance for a small number of nodes, implement a single protection domain and a single storage pool. Ensure that at least two 10GbE ports are available on each node to provide a degree of network-level resiliency; if a single 10GbE interface is used, this will quickly become a bottleneck. More nodes equate to more IOPS in performance, but limited hardware will provide a correspondingly limited amount of performance. Consider that a number of smaller nodes with a few devices may perform better than a larger node with multiple disks; more nodes and more disk devices will provide better throughput numbers than fewer nodes with large numbers of devices. Achieving the maximum utilization of limited hardware resources points towards a hyper-converged system design.

To meet these requirements, a system of 6 nodes, each with 6 HDD drives, configured as a single Protection Domain and a single Storage Pool will deliver the maximum performance given the small amount of hardware. The single SSD device will be used for RFcache.

As rule, caching should be employed to maximize the performance of the system. If there are SSDs available, CacheCade or similar caching solution should be implemented. RFCache should be recommended as well depending on the read/write workflow pattern and the presence of a RAID controller supporting it. Otherwise, the default RAM cache should be increased as much as possible to improve performance.

Figure 19) Small six-node system



In the diagram above, all six nodes are in the same Protection Domain. Each of the six drives in each node belongs to the same Storage Pool, "A" and the single SSD drive, denoted in yellow, is utilized by CacheCade.

DEPLOYMENT II: SMALL TO MID-SIZED SYSTEM

Requirements:

The customer is starting out with a small number of nodes, planning on adding a few nodes in the future. The customer is requesting maximum performance in the form of IOPs and throughput (as opposed to extremely low latencies) and high resiliency, but is trying to limit the total number of SSDs deployed due to their relatively higher cost. Approximately 200TB usable is requested and IOPS in the 30K+ range. No special fault scenarios are part of the requirements. Networking is adequate with multiple ports available for each node in the customer's environment. Cost is a factor; what is the minimal amount of hardware that can be provisioned for maximum efficiency and utilization?

Design Recommendations:

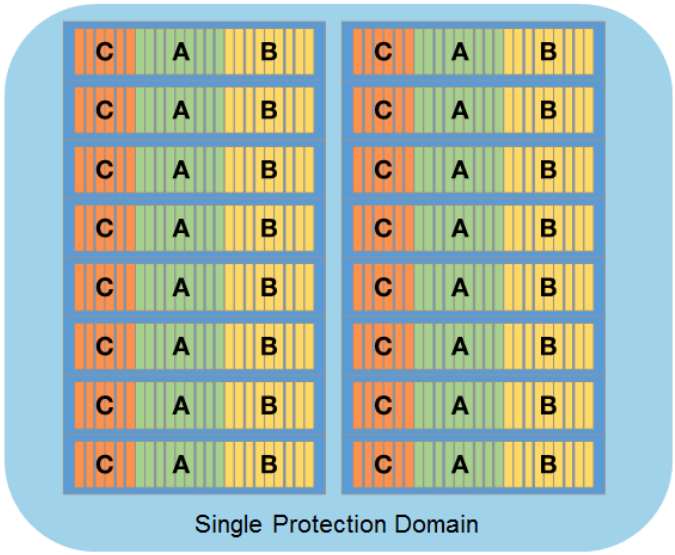
In order to maximize performance for a small number of nodes and devices, implement a single Protection Domain and several Storage Pools for easy expansion. Ensure that at least two 10GbE ports are available on each node to provide a degree of resiliency. Recall that a ScaleIO system requires a *minimum* of three nodes. More nodes generally equate to more IOPS, but cost is a

constraining factor for this deployment. A balance of nodes and devices can be found within the customer's budget. Again, maximum utilization of hardware resources would tend toward a hyper-converged system design.

To meet these requirements, a system of 16 nodes, each with 24 HDD drives, configured as a single Protection Domain and three Storage Pools will deliver 189TB of usable storage and 42K IOPS of performance. In this case, higher performance HDD devices should be utilized.

Because this is a small number of nodes, and the customer plans on further expansion, one Protection Domain and two or three Storage Pools are appropriate. Recall that there is a limit of 300 disk devices per Storage Pools; by starting with multiple Storage Pools, this allows for growth without having to provision additional Storage Pools s and keeps them balanced with respect to the number of devices contained within. Multiple Storage Pools s allow for a balance of performance and protection while allowing for expansion.

Figure 20) Small to mid-sized 16-node system



In the above figure, all 16 nodes are configured identically, with three Storage Pools, "A", "B", and "C" each containing multiple drives per node. If there are SSD drives (consider the drives "C" in the above figure) then they would be provisioned into an all SSD storage pool, "C", or they could be split between a storage pool and use as a caching strategy for HDDs. All nodes are configured in the same Protection Domain.

DEPLOYMENT III: LARGE SYSTEM, MULTIPLE DEPARTMENTS

Requirements:

Customer requires a large system to support multiple departmental environments. Clients must be segregated from each other and not affect each other's volumes. Different tiers of storage providing for different levels of performance should be offered. This system is expected to grow by multiple nodes in each expansion phase. Simplicity in data management is requested.

Design Recommendations:

This scenario pulls in all of the ScaleIO components and utilizes them to provide a tailored system that can serve many different functions simultaneously. The requirement for multiple departments is addressed by implementing multiple protection domains. Each domain can service a different set of clients, isolating performance and capacity from each other while providing security. Different departments can each be assigned a separate domain. Within a protection domain, multiple storage pools can be configured to provide for expansion and delivery of different classes of storage (i.e. SSD vs. HDD). However, a single set of MDMs will be shared across the system.

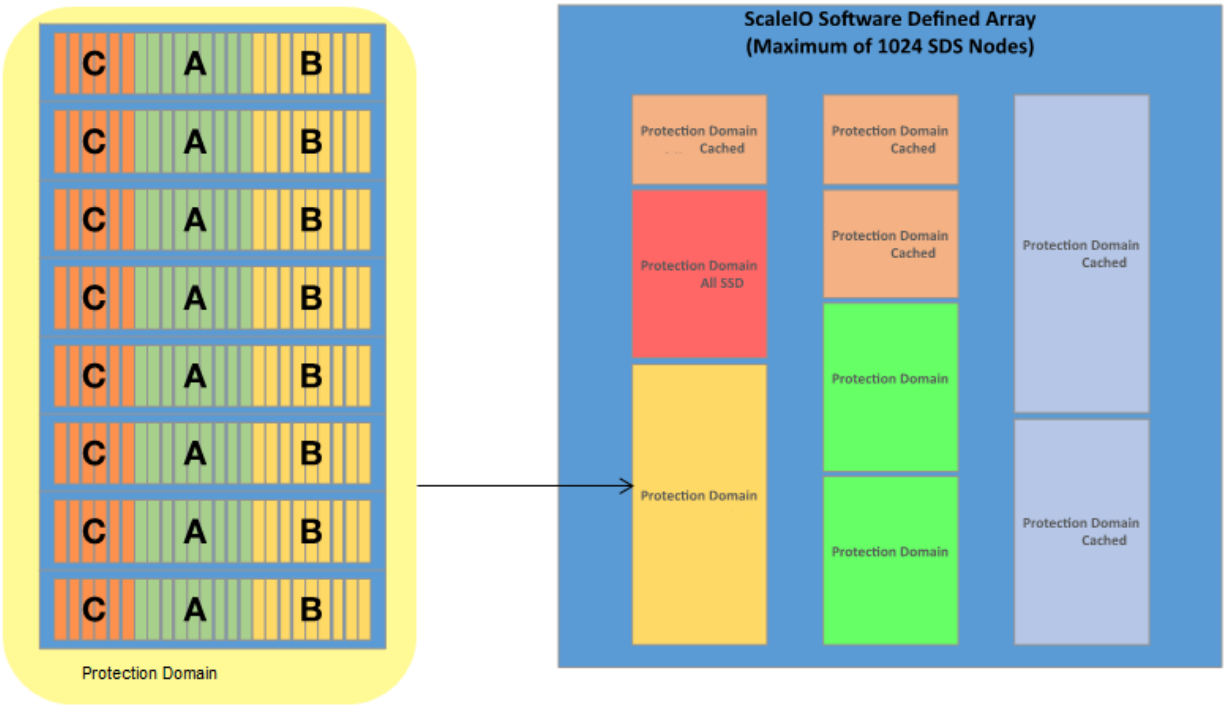
Between ten to thirty nodes should be assigned to each PD, with the sweet spot of approximately sixteen. Nodes should have identical configurations with respect to RAM, HDD, SSD and network interfaces within the same PD.

SSD should be utilized for caching on every node. In addition, SSD tiers from storage pools should be created for volumes with the most demanding performance requirements.

Multiple Storage Pools should be configured to maximize performance, protection, and scalability. Recall that the maximum number of devices per SP is 300; by starting much lower than that, additional nodes with HDD/SSD can be added, gradually increasing the number of devices before hitting the maximum.

By combining many of the concepts outline in this paper, a large ScaleIO system can be created that provides for multiple tenants, each with varying levels of performance needs, all while providing for scalability and expansion.

Figure 21) Large multi-departmental system



In the above figure each of the colored boxes on the right represent a separate Protection Domain. Within each PD, a number of nodes are configured with multiple Storage Pools. These pools might contain SSDs, "C" in the example PD above, for a high-performance pool. Each PD can serve a different department.

CONCLUSION

This guide provides some recommendations for deploying ScaleIO. The best practices and suggestions provided in this white paper are the techniques and settings that has been proven through internal testing and the recommendations provided by ScaleIO engineering and field personnel. After reading this guide, you should be able to understand the concepts of ScaleIO, as well as architect and design a ScaleIO environment.

FACTORS TO CONSIDER

One must consider the specific workflow that the customer is trying to implement on ScaleIO storage. Understanding the workflow that customer is trying to implement is crucial when determining what the expected values are in terms of protection and performance. A ScaleIO system can be configured in a vast number of ways to balance protection and performance depending on the implementation, and so it's necessary to ask questions and get the best available answers before implementation planning:

- Would the workload and application be better served by maximizing availability, or maximizing IOPS?

  Consider five 9's vs. six 9's availability and the required changes in to system design to achieve this.

- Would administrators and stakeholders be better served by partitioning the system to improve availability, or by consolidating to improve manageability?

Improve application availability through the use of multiple protection domains, fault sets, or even through implementing multiple systems. Consider the balance between ease of use/administration of a single system vs. having to manage multiple "partitions"/systems.

- Does the set of workloads and applications to run on the system need a multi-department environment, or a single storage domain?
- What is the most reasonable unit of failure for the system?

  A node, chassis, or rack are all answers that influence storage pool and fault set implementation to provide maximum resiliency against drive (or node) failure. Refer to the previous content on fault sets for examples.

Gathering as much information as possible from the customer prior to deployment ensures that the system is designed and implemented with as many features tuned for an appropriate environment.

Although this guide attempts to generalize many best practices as recommended by ScaleIO, different conditions and considerations apply in every environments. For a specific use cases, it is recommended to contact a representative from EMC, VMware or ScaleIO.

SUMMARY OF SCALEIO DEPLOYMENT BEST PRACTICES

Protection Domains

Use Protection Domains to partition a ScaleIO system:

- When performance isolation is required
- To control where data is stored in a multi-tenant environment
- To increase overall resilience by allowing the ScaleIO system to isolate and recover from multiple failures. more failures

Storage Pools

- Storage Pools should contain only one type of storage device (HDD / SSD), and all devices in a Storage Pool should be the same size and have the same performance characteristics.
- Larger Storage Pools are capable of more I/O concurrency and throughput, smaller storage pools are more resilient and less likely to experience contention; select a Storage Pool size appropriate to the application's requirements.

Data Protection

- Have a scheduled backup policy and disaster recovery plan in place; ScaleIO mesh mirroring does not mitigate these storage protection requirements.
- Keeping the fault unit size to one SDS is a common starting point. Deployments with rack-based failure scenarios may benefit from fault sets.

System Networking and Bandwidth

- If necessary, you can limit SDC IOPS or bandwidth in order to prefer storage system I/O during rebuilds and rebalances over client application I/O.
- When deploying ScaleIO in a leaf/spine network topology, take care not to introduce network bottlenecks between nodes accessing different leaf switches.
- Since low network latency has a positive effect on performance, if possible SDS and SDC nodes should be located on the same network subnet.
- Before deploying applications on a ScaleIO system, use the SDS network test to establish baseline performance characteristics between nodes.
- Use jumbo frames to increase network performance up to 10% when you can be sure that the entire server to client attached network infrastructure supports and is configured to allow them.
- Deploy two or more network interfaces per SDS and SDC whenever possible to increase resiliency and eliminate a single point of failure at the network port level.

MDM and Tie-Breaker

- If ScaleIO system size permits, MDM and Tie-Breaker nodes should be deployed physically separated (such as in different racks) for an incremental, easy increase to system resiliency and availability.
- MDM and Tie-Breaker nodes should be located on the same subnet.

Installation

- When using the ScaleIO system to host VMDK files, select this option during installation.
- When using integrity checking scanning and/or RecoverPoint software, Zero Padding must be enabled.

- ScaleIO will enable data redundancy via mesh mirror and no host-level RAID or mirroring is necessary.

Caching

- Do not enable all cache mechanisms on the same SDS node at the same time. RAM Read Cache and RAID-controller-based write-back cache with non-volatile memory like CacheCade may be used simultaneously.
- If a server has a hardware RAID controller, use it to improve read and write performance if it is battery-backed; otherwise, only use it as a read cache.
- RAM Read Cache should not be implemented on SDS nodes hosting SSDs.
- Tune memory buffer settings according to the number of storage devices attached to an SDS.

System Design

- Whenever possible, SDS nodes within the same Storage Pool should be provisioned with identical components and storage devices.

PRODUCT LIMITS

ITEM	LIMIT
ScaleIO System raw capacity	300 GB — 16 PB
Device size	100 GB — 6 TB
Minimum Storage Pool capacity	300 GB
Volume size	8 GB — 1 PB
Maximum number of volumes/snapshots in system	32,768
Maximum number of volumes/snapshots in Protection Domain	32,768
Maximum number of volumes + snapshots in single VTree	32
Maximum capacity per SDS	64TB
SDSs per system	1024
SDSs per Protection Domain	128
Maximum devices (disks) per SDS server	64
Maximum devices (disks) per Storage Pool	300
Minimum devices (disks) per Storage Pool	3, on different SDS instances
Maximum SDCs per system	1024
Maximum volumes that can be mapped to a single SDC	8192
Maximum Protection Domains per system	256
Maximum Storage Pools	1024
Maximum Storage Pools per Protection Domain	64
Maximum Fault Sets per Protection Domain	64
Maximum SCSI Initiators per system	1024
Maximum IP addresses per server (MDM and SDS)	8
RAM Cache	128 MB — 128 GB

REFERENCES

- EMC ScaleIO for VMware Environment Deployment and Performance Best Practices

  http://www.emc.com/collateral/white-papers/h14684-scaleio-vmware-bpg.pdf

- Introduction to XtremCache

  https://www.emc.com/collateral/white-papers/h11946-introduction-emc-xtremsw-cache-wp.pdf

- EMC ScaleIO Networking Best Practices

  http://www.emc.com/collateral/white-papers/h14708-scaleio-networking-best-practices.pdf